EXTRA! EXTRA!
IRVINE IS BURNING

WITH EXCITING NETWORK RESEARCH

Record of the
***5th Workshop on Hot Topics in Networks**:*
HotNets V
Beckman Center, University of California, Irvine
Irvine, California, USA
November 29–30, 2006

Record of the Fifth Workshop on Hot Topics in Networks: HotNets V

—

Beckman Center
University of California, Irvine
Irvine, California
November 29–30, 2006

—

Sponsors



ACM SIGCOMM

US National Science Foundation
for student scholarships

—

General Chair

Xiaowei Yang — *UC Irvine*

Program Committee

Frank Dabek	*Google*
Cristian Estan	*University of Wisconsin, Madison*
Anja Feldmann	*TU München*
Rebecca Isaacs	*Microsoft Research*
Eddie Kohler	*UCLA* (Co-chair)
Ratul Mahajan	*Microsoft Research*
Greg Minshall	*Unaffiliated* (Co-chair)
Vivek Pai	*Princeton*
Vern Paxson	*ICSI/LBNL*
Sylvia Ratnasamy	*Intel Research Berkeley*

Steering Committee

Larry Peterson	*Princeton* (member to 2006)
Scott Shenker	*ICSI and UC Berkeley*
Alex C. Snoeren	*UCSD*
John Wroclawski	*ISI and MIT*

Contents

—

Forward

—

Welcome to the 5th Workshop on Hot Topics in Computer Networks!

We received 114 submissions. The ten members of the program committee submitted 421 reviews, and external reviewers submitted an additional six reviews. HotNets papers are shorter than a normal conference submission, and were generally enjoyable reading, but the load on the program committee was still non-trivial, and they all deserve our thanks. Additional thanks are due to our external reviewers: Kevin Fall (Intel Research Berkeley), Philip Levis (Stanford), Nikitas Liogkas (UCLA), Richard Mortier (Microsoft Research), Suman Nath (Microsoft Research), and Scott Shenker (UC Berkeley and ICSI).

Of course most thanks are due to the authors of submitted papers, both accepted and rejected. Thank you for sharing your work with us.

We would also like to thank our general chair, Xiaowei Yang, for her work in organizing the venue, proceedings, student travel grants, and so forth for the workshop.

We hope you find the papers in this workshop stimulating, and we expect two days of interesting conversation.

Paper analysis Now to the data. Figure 1 lists a number of discriminators for HotNets papers, ranging from review scores (1 was low and 5 was high) to selected topics and assigned reviewers. For each discriminator, we list how many papers fit that discriminator, and what percentage of those papers were accepted. Future authors may want to scrutinize this list to increase their chances at the next HotNets. Consider writing a theoretical paper that claims to be about no other topic (neither systems, routing, transport, security, robustness, nor link-level or wireless issues), and make sure it interests Cristian Estan. And above all, keep it short.

The program committee agreed more on accepted papers than rejected ones, and more on novelty and discussability than overall merit. Define the *spread* as the difference between a paper's highest score and its lowest score in a given score category. (There were four categories: overall merit, technical merit, novelty, and discussability.) Then the set of rejected papers had in every category higher average spreads than the set of accepted papers. This was dramatically true for novelty and discussability, where the average spread for rejected papers was roughly 60% higher than for accepted papers. Did we generally prefer lack of controversy (unfortunate for HotNets)? Were reviewers with outlier scores more successful at convincing the program committee to reject a paper than they were at convincing the committee to accept one? Or are good papers obviously good to most everybody?

Discriminator	**# Papers**	**Accepted**
Low "overall merit" ≥ 4	5	100.0%
High "novelty" $= 5$	4	100.0%
High "overall merit" $= 5$	9	66.7%
PDF submission < 100000 bytes	24	41.7%
Topic "None of the others"	5	40.0%
Topic "Theory"	6	33.3%
Reviewed by Cristian Estan	34	32.4%
Topic "Security & robustness"	27	29.6%
Mainly Computer Modern fonts	7	28.6%
Reviewed by Vern Paxson	34	26.5%
Reviewed by Sylvia Ratnasamy	34	26.5%
Non-anonymous submission	85	24.7%
Topic "Controversy"	21	23.8%
Reviewed by Frank Dabek	34	20.6%
Reviewed by Greg Minshall	113	20.4%
Any paper	*114*	*20.2%*
Reviewed by Eddie Kohler	35	20.0%
Reviewed by Ratul Mahajan	36	19.4%
Reviewed by Anja Feldmann	34	17.6%
Topic "Systems"	46	17.4%
Topic "Applications"	21	14.3%
PDF submission ≥ 500000 bytes	14	14.3%
At least one TrueType font	31	12.9%
Reviewed by Vivek Pai	33	12.1%
Reviewed by Rebecca Isaacs	34	11.8%
Topic "Routing & transport"	37	8.1%
Anonymous submission	29	6.9%
Topic "Link & wireless"	26	3.8%
High "overall merit" ≤ 3	54	0.0%
High "discussability" ≤ 2	23	0.0%

Figure 1: *How to get your paper into HotNets V, or not.*

We were more willing to give low scores for overall merit than for technical merit or novelty. For several papers all reviewers agreed to reject, but no paper had all reviewers agreeing on "no technical merit".

Figures 2 and 3 rank the papers in increasing order by overall merit, technical merit, and novelty, then plot novelty rank vs. overall merit rank (Figure 2) and novelty rank vs. technical merit rank (Figure 3). Novelty

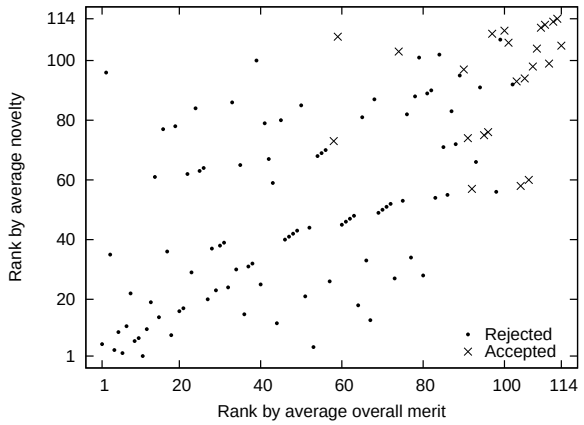


Figure 2: *Novelty vs. overall merit. We largely accepted papers that had either high overall merit or high novelty.*

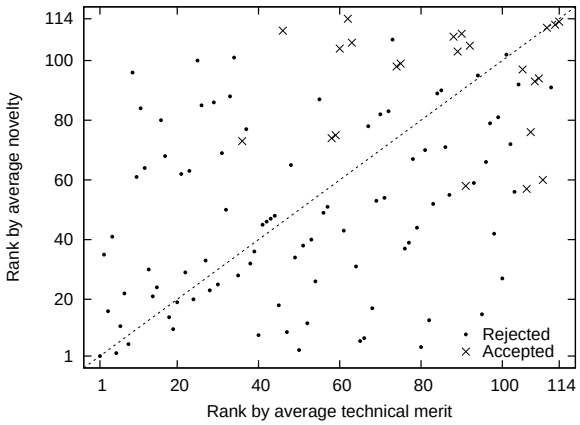


Figure 3: *Novelty vs. technical merit. We preferred "exciting, but flawed" papers, with higher novelty than technical merit (13 acceptances above the line), to "boring, but correct" papers, with higher technical merit than novelty (7 below it).*

rank is a better predictor of acceptance than technical merit rank, as it should be here. Rank plots of both novelty and technical merit vs. overall merit cluster near the diagonal, implying, as one would hope, that overall merit is correlated with both factors. Figure 3 shows less diagonal clustering, particularly at the high end. It's hard to be both new and right.

Anonymity HotNets V authors could choose whether or not to submit their papers anonymously, and just over one-fourth of authors chose to do so—fewer than we expected. Anonymous submissions were much less likely to be accepted than non-anonymous submissions. Of the discriminators in Figure 1 that are under the author's control, only the topic "Link & wireless" is correlated with a lower acceptance rate. This is also reflected in the overall merit scores, where anonymous submissions did substantially worse. Rejected anonymous submissions came from top worldwide industrial research labs and top US universities, among other places. Were our referees less likely to give anonymous submissions the benefit of a doubt? Or, perhaps, were submitters less likely to put their names on a less-than-top-quality submission?

HotNets V *reviewers* could also choose whether or not to submit their *reviews* anonymously. Both co-chairs planned to submit their reviews non-blind, allowing authors to see who wrote their reviews, and both co-chairs still believe philosophically in open reviews. In the end, however, only two people—Greg Minshall and one of our external reviewers—chose to submit their reviews non-blind. The other co-chair, Eddie Kohler, had wanted to use open review as an enforcement mechanism to ensure his reviews were careful, very high quality, and avoided unnecessary provocation, but wasn't sure they all met that standard in the end. If anything, this supports the argument for open reviews, but when even workshops like HotNets receive 114 submissions it's hard to know where all reviewers would get the time to complete their reviews that carefully.

Onward And with that, enjoy these papers: 2006's glimpse into the future of networking research!

—Eddie Kohler and Greg Minshall

From the Reviews

—

What follows is a lightly edited selection of interesting, supportive, and contrary tidbits from the program committee's reviews of the papers selected for HotNets V. The first, italicized paragraph summarizes the paper. The editing has conflated comments made by different program committee members, so "I" may refer to a melded PC hive mind rather than an individual. Of course, reviews reference the *submitted* versions of the accepted papers. The authors have addressed some, but not all, of our comments in their final copies; it's interesting to see which comments led to revisions. We hope you enjoy this look behind the curtain.

(R)Evolutionary Bootstrapping of a Global PKI for Securing BGP

—

Develops a scheme by which ISPs can develop a global PKI for authenticating routing updates in an incremental *fashion. The key trick is based on ISPs first issuing self-signed certificates which other ISPs honor based on earliest-seen-cert. Thus, an attacker can undermine an ISP's certificate only by issuing a (misleading) cert prior to the ISP issuing a cert. This both gives ISPs an incentive to issue certs and requires attackers to act quickly as the rollout proceeds.*

You might face a problem in convincing the global players to actually agree on a grassroots certificate authority. If the routers can work with multiple roots of trust, why would ISPs pressure the emerging roots to combine into a single root? I understand that less trusted roots will want to group under a trusted single root, but the most popular/trusted CA roots might have limited willingness to do so. Still, even the global players are likely to prefer a grassroots certificate to no certificate. Indeed it is a "nice" feature that you can force this by finding someone evil. It's a nice gimmick that evil attacks can be resolved by increasing the security level!

The approach depends on the premise that attackers will be ultimately detected, which is worrisome, particularly if attackers confine their activities to a limited window of time. But even though "first cert is legit" is brittle, particularly as attackers become aware of it, it also has the property of mapping to how things are often deployed (i.e., legit use predominates initially).

Do prefixes get hijacked intentionally, or inadvertently, via misconfiguration? If the latter, the security of private keys would seem less of an issue.

Don't Secure Routing Protocols, Secure Data Delivery

—

Argues that connectivity (or availability) is a much simpler problem for routing to solve than path integrity (i.e., that the connectivity is achieved via the correct set of routers), since endpoints can verify connectivity. Therefore, routers should be focused on providing as many connectivity options to a destination as possible ("Availability Centric Routing", or ACR) rather than trying to achieve an optimal path or trying to ensure that malicious actors cannot manipulate the path. A combination of ISP-assisted source routing, end host-based measurements of path characteristics, and cryptography could secure the current Internet against route hijackings without the deployment of any flavor of secure BGP (or other changes to routers) as long as the tier-1 ISPs are trustworthy.

An interesting thought experiment that directly frames my own wondering about secure routing.

ACR introduces high costs for end systems. There's the overhead of locating the route; shouldn't that be the task of the ISP's? It seems a bit strange that you seem to imply that it is easier for security-unconscious end users to secure *all* end systems, than it would be to secure a major part of the network infrastructure that is actually administered by folks that should know at least the basics of security.

If we had secure routing then we could be more lax about end-to-end integrity checks; the paper should acknowledge that this is something significant we give up (though admittedly relying on routing to give us address-based authentication is brittle).

This paper makes perfect sense and it is totally contrary to conventional wisdom (and on top of it all it has a reasonable path to adoption). Looks like the perfect HotNets paper!

On the other hand, regarding conventional wisdom, the paper could more explicitly bring out the connection between the premise and the end-to-end principle.

A Technical Approach to Net Neutrality

—

Proposes a technical solution to the network neutrality problem. ISPs favoring net neutrality would deploy in their networks neutralizers that would have the role of obscuring from discriminating ISPs the addresses of content providers that send traffic through discriminating ISPs.

Creative, topical, and thought provoking, but I can't fig-

ure out if it is hasty or thoughtful. Either way, it opens up an interesting line of research.

The central assumption that some ISPs will actively participate in your proposed scheme seems a little suspect—it's one thing for an ISP to speak out in favor of network neutrality but quite another for it to actively assist its users in circumventing the techniques put in place by other ISPs (some of which it might have business relationships with!). Seems like this might justify ISPs retaliating with attacks—some would say defenses—aimed at the neutralizer mechanisms.

While the approach is different than that for anonymizers, it's similar enough to them, and to VPNs. Basically, it's pretty much the solution one pictures arriving at given the formulation of the problem, with the mildly innovative aspects being the use of statelessness and then leveraging this to use anycast. I don't buy that the ISP can no longer discriminate. It can if it knows who the neutralizer's customers are and is okay with discriminating against all of them en masse. Note that this means it's difficult for anyone to run a neutralizer except on a large scale with a lot of customers.

Presumably only at-risk traffic uses the neutralizer service. In that case, why wouldn't just discriminating against all neutralized traffic be fairly effective? If market forces could indeed be relied on to discourage ISPs from such discrimination, then wouldn't network neutrality be less of an issue?

You take a narrow view of net neutrality and discriminatory behavior, in which an ISP wants to disfavor one particular content provider in the hope of extorting money from it. There are other goals and discriminatory behaviors that you don't try to prevent. For instance, a discriminatory broadband ISP may favor one content provider, maybe because it pays extra money; in a world where this content provider has very few competitors, this has the same effect as disfavoring the remaining, which is something that you were trying to avoid. Other sophisticated "attacks" against your proposal are also possible, such as examining the nature of the traffic generated by an application. For instance, it is probably trivial to identify VoIP traffic if I wanted to discriminate against it. Trying to prevent such attacks, for instance by normalizing traffic of all applications, would significantly increase the cost of the solution (and may hurt some applications). This is a form of packet discrimination that you do not consider.

Hoping that consumers are in a strong position to make changes may be true in the long run, but that is the long run.

Wouldn't regulation, when it finally comes about, trump any technical solution?

An Axiomatic Basis for Communication

Argues that the field would benefit from a formal framework by which to formulate and reason about networking tasks. It therefore develops a framework for formalizing network communication similar to that of denotational semantics for formalizing program execution. The framework includes an interesting set of definitions and primitives that capture the operation of a forwarding element. These are then composed to reason about activity along a multi-hop path.

The high-level goal here is great. The value is in considering that there might be an apt, formal set of principles, although this particular approach might not have the right set of axioms. Very flawed, but provocative and in an interesting direction.

The formulation focuses mainly on simple message delivery. Of networking tasks, this seems like the most easily amenable to this form of analysis: routing is a standard algorithms concept with well-defined state variables, and one can reason about overall correctness in terms of the consistency of just this state. That is, one can reason about deliverability in terms of just routing entries that effectively abstract away all the computation—BGP, policy, intra-domain, etc.—that went into computing that state. Many networking tasks—congestion control, reliability, security, latency, etc.—seem to lack this property. How useful will this approach/formulation be for such tasks? And given that the very simple NAT example is > 10 lines, how complicated will this get as we try to express even slightly more realistic tasks (e.g., fetching a web page that involves client, proxies, routers, L7 switches, load balancers, DNS resolvers/servers, web caches, server clusters, etc.)? The PODC/DISC/theory community offers many analysis tools and approaches. We don't seem to use them in part (I suspect) because things get too hard when we try to capture the zillion dependencies and interactions in real systems.

The network is (approximately) the one object we consider in "CS" which is not assumed to be fail stop. Additionally, it has phase delays (though we worry about that in large scale hardware designs): state changes take place, if at all, at different times and in different orders.

I like where I think this is trying to go but can't understand the paper.

Protocol Design Beyond Graph-Based Models

Wireless protocols frequently model inter-node communication (or interference) by means of graphs—an edge is placed between two nodes that can directly communicate (or interfere). Graph-based models are widely acknowledged to be wrong, but everyone uses them anyway. This paper adds urgency to the search for better models by showing through measurements and protocol designs that non-graph-based protocols can perform much better than graph-based models in practice.

The call for improved modeling of wireless communication is a very useful one and the empirical results are eye-opening. However, it feels a bit like one camp of theoreticians talking to another camp of theoreticians. Systems have moved beyond graph-based models already; per-

haps the work is a bit late. Nevertheless, people *do* continue to use graph-based models, for example in sensor networks, and theoreticians are people too, and the paper is admirably clearly written—and decorated with useful experiments! Furthermore, the suggested protocols differ somewhat from the prior work I know of. Presenting this work could make a difference.

It could be greatly enhanced if the authors proposed a protocol based on their understanding of wireless networks through physical models.

The paper implies that most protocols are *designed* to fit a graph model, whereas I believe that it is far more common to see protocols being *evaluated* (not designed) using graph models. Are there examples of in-use protocols that globally schedule packet transmissions based on graph-based models? Graph-based models are incredibly simple to use for evaluation purposes. It would be good to include some discussion on what parameter-setting complexity is required in the SINR-based models. Sounds like there is: the relation between power and distance, the value of alpha, the interference at the receiver, noise at the receiver and beta?

Though the proofs about capacity are interesting, the result is underwhelming: the distinction between $<$ and $\leq$.

Some Implications of Low Power Wireless to IP Networking

—

Describes a study of the performance of a personal-area-network link layer, namely 802.15.4, and considers the resulting implications for IPv6 routing.

This paper is well-written, readable, and presents the problems in much more interesting depth than most of the other papers in its class. Reading it, one gets the impression that there are interesting research problems in this space, and that the techniques used by systems like Roofnet are not immediately applicable to this domain. The authors also give some good insights into the tradeoffs in code complexity, network reliability, and power/memory consumption in this environment, which is a much different set of tradeoffs than the wired world. Overall, it's a good eye-opener, and it has the data to support its claims.

The IPv6 connection seems tenuous. Many of the Implications apply to technologies other than IPv6. The two directions of an IP route may need to differ, but even if 802.15.4 nodes have IP addresses, will they route via IP routing? Probably not. And asymmetric routing is common in IP networks anyway.

The nugget I got out of the paper was about acknowledgement losses causing problems for fragmentation reassembly and the ETX metric. Is the lack of intermediate links in your testbed mostly due to its being indoors and anchored? Do you have any evidence that the link technology would cause different observations here? Do you have an 802.11 testbed for comparison?

Network System Challenges in Selective Sharing and Verification for Personal, Social, and Urban-Scale Sensing Applications

—

Articulates privacy and accuracy concerns that should be addressed by future applications that rely on sharing sensor data in personal, social and urban settings. A distinguishing feature of these applications is that the sensing devices, ranging from tiny motes to expensive video cameras, are owned and operated by individuals.

The application space the authors highlight is one that has been receiving plenty of attention, and deployment, in certain communities (HCI/ubiquitous computing, wireless), and we're probably overdue for a networking paper on the topic. While the paper offers little new information, it does a nice job in pulling the discussion together and identifying the challenges, and could serve as a useful starting point for a discussion on the network implications of personal and urban computing.

While the straw-man architecture is useful, it's a bit of letdown in that it's mostly a formalization of how these applications are built today—i.e., a sensor network relays data to a server through a few proxies and clients query the server, again through a bunch of proxies. This proxy-centered architecture is traditional and somewhat limited; scenarios of a large number of mobile, autonomous, and yet related sensors (e.g., camera cellphones in the same city block, building) are at least as important and probably more challenging. Much of the novelty in your architecture seems to be in getting the selective sharing and context verification right, but it isn't clear how much of this is a networking issue, and the paper doesn't explore this in much depth other than telling us where in the infrastructure such functionality would be implemented. It would be nice to see a straw-man of data naming schemes, query language and pub-sub interfaces that might support some sample sharing and verification policies.

Are there organization/provider boundaries that need to be respected? Is this easily done with a DNS-like infrastructure that assumes an administrative hierarchy? What is the business relationship between client/sensors and their mediators, are there charging/accounting requirements? You say mediators are like firewalls in various ways, but if I am behind a firewall, I know who runs the firewall, and can hire and fire that person. I don't get the sense that mediators have that level of administrative "closeness" to their users.

"Citizen" *is* an odd word, but I have no alternative.

Rethinking Wireless in the Developing World

—

Describes efforts to connect rural communities in Africa and India via long-range wireless links.

I enjoyed reading this paper. While folks have been working in this area for the past few years, this is the first time I have seen anything close to a research agenda laid out, and the discussion about density and WiLD (wireless long distance) vs. mesh is also very interesting.

Remote upgrade and management seems key to running these networks at a low cost, but your story on that front seems pretty weak so far. If there are existing satellite and GSM connections that you can use, why bother deploying WiLD? Presumably, the deployment cost for such networks has already been paid by someone, and just using them (more or less) shouldn't be that expensive.

High-gain directional antennae have not taken off because of the management problems they bring about in the face getting disoriented (e.g., during high wind conditions). Do you face this problem? Electronically steerable antennae, which you also propose for fault tolerance, can mitigate this. How expensive are such antennae?

The authors argue for long-range wireless (instead of cell, satellite, etc.) based mainly on the lower cost of deploying commodity wireless equipment, then argue that WiFi is a better choice than WiMax due to the latter's cost. This is an interesting argument and I would like to see it better explored: it's a bit surprising that a technology designed for connecting users in the same room via a base station is a better way to connect users over a wide geographic area than a technology designed to, well, connect users over a wide geographic area. How much does the spectrum necessary to run WiMax cost in Ghana? How would WiMax or microwave relay perform at 50 km? Also, cost is repeatedly cited as an overriding design decision; it would be nice to know how low the budget actually is. The project isn't well-funded enough to upgrade a 256 MHz CPU?

The proposed changes to the 802.11 MAC aren't very carefully examined or justified; data from the real deployment or even a mention of how well the real deployment actually performs would be helpful. How many of these problems are the result of choosing WiFi over other technologies? Does WiMax's MAC address many of the MAC issues described here, such as stop-and-wait?

Service Portability

—

Use HTTP redirection to find what you really want! The prototype Permafind system enables service portability, e.g., for email, blogs, web pages, and so forth, using the standard mechanisms of redirection, indirection, relaying, and proxying. It is intended as an incremental solution that is immediately deployable.

This paper amusingly describes Permafind as "technically boring", and yes, it doesn't hold any huge technical innovations. How can you not like a paper that says right in the abstract that it has no technical innovation? However, the pragmatic combination of simple mechanisms to solve a pressing problem is a major strength. It is refreshing to encounter a well-thought-out solution that seems to actually achieve its own goals. This may actually end up being deployed on a large scale, although perhaps not in the form the authors envisage, because it is simple, useful and dare I say it, obvious. Although I don't like the solution in the long term, in the short term, and to inform the long term, it's nice, simple, and cute.

It seems to me that Permafind is itself a provider, so in solving the problem of provider changes, you add a new one.

I didn't see much of a discussion of problems with this approach, such as how to handle bookmarks, or the problem with search engines giving the ephemeral pathnames instead of those findable through the redirection service.

The End of Internet Architecture

—

The main argument is that the basic notion *of considering possible network architectures for a future Internet is detrimental, as it presupposes that there should be a distinction between nodes that use the network vs. nodes that facilitate the network's operation. That is, the idea of a "network architecture" imposes an artificial distinction between "networking" and "distributed systems" that needn't be fundamental and erroneously presupposes the nature of how a future global networking infrastructure should be structured. In summary, network architecture is pernicious and should be stamped out.*

The author's emphatic decrying of the damage done by imposing a separation between networking and distributed systems doesn't resonate for me. I don't see that separation as particularly manifest, and to the degree in which I do see it (as evidenced in the network systems papers that appear in SOSP vs. SIGCOMM, say) I don't see it as rooted in the notion of network architecture so much as in how networked systems emerged from multiple parent disciplines (operating systems, digital communication). Furthermore, to me, the term "network architecture" means "a set of networked communication abstractions and the relationships between them", whereas I eventually gleaned that for the author it more specifically means the layered, hosts-vs.-routers structure that *today's* Internet manifests. Surely it's clear that a future architecture needn't impose that style of structure, although it *will* need a set of abstractions along with (coherent) relationships between them.

My model of systems, whether architectures or large software systems, is that once you deploy it (assuming it gets used a lot) it starts to accrete barnacles. After some period of time, you re-do the system from scratch (hopefully), incorporating as many of the barnacles as seems right into the new design (so now they become layers, or compartments, or functions, . . .), and deploy it. At which point, it starts to accrete barnacles.

Suppose we do as you propose: what is the guarantee that the result of this organic growth will be a desirable state?

Sometimes a paper I agree with makes me question my own beliefs. This is one such paper. For the most part, I agree that network architecture is another name for software, and it's nice to see that point expressed. But in practice there is one important difference. Network architecture specifies packet headers. These well-specified headers, and the behavior behind them, add to the ways in which the network may be programmed. Only a tiny fraction of the libraries in existence (C? C++? MFC? POSIX?) are specified anywhere nearly as compactly and effectively as the Internet protocols. Those clean specifications of *data*—what's on the wire—have helped us add new functionality without breaking existing networks. Thinking about headers is different enough from thinking about software that it might deserve a different name. Not maybe what conventionally people mean by network architecture, but still.

While I think this paper will provoke discussion, I don't think it'll be particularly illuminating discussion. It'll be more like a discussion between parties who spend most of the time arguing until they figure out that they mean different things by the same terms.

The paper feels as if you think it's shocking, but in practice I think most researchers don't care that much about network architecture.

Cut Figure 1. It's so bad it transcends bad.*

DECONGESTION CONTROL

—

The paper turns our viewpoint on managing congestion upside down: rather than design mechanisms that worry about picking a conservative transmission rate to avoid packet drops, the authors propose that hosts simply transmit greedily at a high rate and routers instead drop packets fairly across active flows. To tolerate drops, the authors propose that endhosts erasure code their data streams.

An intriguing idea, and definitely novel, but the authors could have gone further in trying to convince us that it might actually be sensible.

The impact of dead packets sounds like a possible showstopper. If it turns out that congestion is not just limited to the edges of the network (inter-continental links?), then should we just give up on the idea of decongestion control? Is there some fix one could come with, or if not, how bad might the wastage be?

You say that "end-host congestion control is typically suboptimal and, critically, relies on the goodwill of end hosts for success." Well, any practical control protocol for something as complex as the Internet is going to be "suboptimal", at least according to some metric. Second, is this problem with nice/evil hosts a problem in *practice*, or just in theory?

The TCP receive window doesn't *ensure* that the sending rate doesn't exceed the receiver's ability to consume data, it just makes it very very likely. Secondly, since Van Jacobson's 1988 paper, it hasn't been clear that we actually need window updates. Why do we "privilege" end hosts over routers (who have not so much control over the load dumped on them)?

What exactly is the difference between current TCP and the proposed protocol? Eventually, there are two proclaimed differences: no retransmissions of lost packets due to the use of erasure codes, and novel greedy congestion control. But you have suggested a schema where the caravan size and type can be adopted. How does this differ from adopting the cwnd? Isn't it possible to game the proposed system by ignoring the feedback? Isn't it possible for an aggressive sender to just send with a huge redundancy and thus succeed?

Is this more or less incrementally deployable than (say) XCP? And given current router capabilities, is the per-flow state due to fair queuing really an issue?

A SIMPLE APPROACH TO DNS DOS MITIGATION

—

A simple approach to improving the client-perceived reliability of the DNS system: DNS servers are allowed to aggressively cache records past the records' TTLs in a "stale cache." A record in the stale cache can be used only when the authoritative servers for the record's domain are not available; the stale cache allows queries to be answered despite server failure.

This paper is great: a simple solution that appears to do the trick. I agree that the new DNS architectures are likely an overkill, given that DNS seems to work the vast majority of the time. Your arguments about your approach making the right trade-offs are convincing. Your easily deployable change to DNS resolvers that can make a real difference in some cases. The weakness is that the proposal is so simple and obvious (in hindsight) that it will probably not generate much discussion.

A true DNS DoS defense would allow the DNS server to serve updated information about the domain(s) it is authoritative for. Something along the lines of "mitigating the effects of DoS attacks on DNS servers" might be more fair.

I don't like not knowing when (any) state in network will be purged. I used to work for a CEO who said he would sign (almost) any contract, as long as he could get out of it in a finite period of time.

I suspect this will be effective in practice, but it is hard to say because of the underlying Zipf popularity distributions. On the positive side: the resolvers are likely to have in their stale caches information about popular zones; the less popular zones may be missing but they are also less likely to be attacked (?). On the negative side: if servers higher up in the hierarchy, e.g., the root servers, are attacked, many unpopular zones under them will be unresolvable because they will not be present in the stale cache; the heavy tail may imply

*In the final copy, Figure 1 remains.

that a significant fraction of queries are for zones that have not been resolved in the past.

Does your scheme interact poorly with diagnosing local DNS-related problems? Consider a situation where I can reach my local resolver which cannot reach other servers due to some problems. Today, I will discover this quickly because I'll not be able to resolve most domains. But with your approach, I will continue to resolve (to stale information) many of the names without realizing that something is amiss.

SPACE: Secure Protocol for Address Book Based Connection Establishment

A conceptually simple but nice insight regarding a way to leverage easily-discovered evidence of likely trust alignment, coupled with modest use of an associated, out-of-band secure channel. In the realization of this insight, two users who want their PDAs to associate first engage in a protocol that allows them to verify whether the other user is already in the address book stored in their PDA (i.e., telephone numbers, perhaps street addresses, etc.). If so, then they use the contact information already in the address book to perform a key exchange, for example by sending a text message to the cell phone number in the address book. This second step means that an attacker who lies about their identity won't be able to complete the protocol unless they have also compromised someone's out-of-band access.

The paper includes an extensive security and privacy analysis, but doesn't particularly explore the issues of (1) latency in establishing an association, (2) how often will one want to establish an association with someone not in one's address book, (3) the requirement that address books have a canonical form to enable correct matching.

I understand the general problem that this work is trying to solve, but by restricting it to the set of people who have each other in their address books, it makes the problem either a lot simpler or trivial, depending on how you look at it. In considering the problem, it also makes sense to consider the alternatives. I guess one could install the necessary software on both ends to allow this kind of key exchange. One could also just include public key information in vcards and include them in address books.

The described protocol is unnecessarily complicated considering the bottom line: it's only as secure as sending an SMS message. For example, what does the first phase (where contact information is exchanged) provide besides the phone number we'll use to send the SMS containing a public key? Are there simpler possibilities that are equally secure? Since Alice and Bob exchanged contact info at some point in the past, they could have exchanged a public key; or, if users aren't good about maintaining keys, why doesn't Bob just SMS his public key to Alice immediately? If the key comes from a phone number in Alice's phone book she'll associate that key with the entry. Neither SPACE nor these straw-man protocols are very satisfying considering that the security "bottoms out" in SMS. In contrast, existing Bluetooth authentication techniques, though currently cumbersome, have a refreshing basis in physical reality. For example, when inputting a PIN to bind my Bluetooth keyboard to my desktop the chain of trust is short and ends with me trusting my OS and display, not an unknown telecom network's SMS system. A system that was more automatic, but with the same easy to grasp security guarantees, would be more compelling.

One attack not discussed is theft of a PDA which is also the user's cell phone, enabling the attacker to complete the out-of-band part of the protocol.

The idea is fairly modest, but I liked the basic insight.

Exploiting Social Networks for Internet Search

Presents a search engine that indexes locally cached content. This lets it take advantage of local context and annotate Google results with locally relevant results. Essentially, users of a social group help enhance Google's web search results by sharing each others' indexed browsing history.

Does a good job of showing a new direction where systems researchers can make progress on the challenging and important problem of better web search. I found the paper non-obvious, quite intriguing, and promising in terms of the possible performance gains.

You might think a bit about k-anonymity. Hits in Google don't tell me much; hits in a local cache tell me, for example, that a male colleague is looking to date women in San Diego. How to allow a user to share useful information, without exposing themselves to leaking personal or private information, is a challenge.

Don't we all know how to refine our search by adding terms to find the relevant page?

Free Riding in BitTorrent is Cheap

Discusses a way to build a free-riding BitTorrent client, and shows by actual experiments that this client can maintain healthy downloads without uploading anything or uploading faulty items. Since there is a general impression that BitTorrent is good at hindering free-riding and is robust to attacks, the paper shows an interesting contradiction.

You show that having many open TCP connections does not harm performance, as commonly believed in the BitTorrent community. Maybe you could delve a little bit more into why that is the case, given the fact that BitTorrent has chosen to maintain only a few open connections, supposedly for better performance. You also show that, even when downloading only from leechers, the performance of a selfish peer is still acceptable, a remarkable result.

The idea that such free-riding attacks can be used by corporations for fighting uncontrollable distribution of copy-

righted material is indeed a new one, and deserves more attention.

It comes as no surprise that the BitTorrent protocol can be gamed. Not only have selfish attacks been demonstrated before, but intuitively one would expect the protocol to be vulnerable. The incentives built in to BitTorrent are not ironclad, they are not designed to be. If a freeloader downloads slower than a non-attacker, who would want to be a freeloader? BitTorrent incentives provide "economic" *encouragement* to upload. They are not meant to *strictly require* uploading; there is no security consequence if a freeloader can download without uploading. If you could download *faster* without uploading, that would be a big deal, and might cause BitTorrent to fail eventually, but it seems from your results that no one would freeload in practice, yes? If this isn't true, then why not? So it's not clear whether "lack of punishment ... raises concerns about the future of peer-to-peer file sharing".

That a "sharing community" is so much easier and more advantageous to attack is intuitively obvious, but only after reading the paper.

Capturing Complexity in Networked Systems Design: The Case for Improved Metrics

—

Proposes a quantitative measure of the complexity of network algorithms that would capture the intuitive qualities of "cleanness" and "elegance". In summary, the paper attempts to quantify networked system design taste.

I found this proposal highly stimulating. I started out thinking that the problem was obviously intractable, but came away with a sense that the idea really could go somewhere. This was the most unexpected paper in my set, yet it has the feeling "why hasn't anyone done this before?". It seems both intuitive and the correct approach, and tackles a problem that is both very important and has received little rigorous attention so far.

How might metrics be used to capture notions like "robustness"? Can they illuminate the tradeoffs of hard vs. soft state (briefly alluded to at the end), or benefits from adhering to the end-to-end principle?

Thinking about complexity is important and interesting. However, I'm not sure this paper doesn't just substitute subjective aesthetic judgments about metric formulae and parameter values for current subjective aesthetic judgments about protocols themselves.

There is one deep way in which the intuition codified in the paper differs hugely from my own, which is that "complexity decreases as the network offers more delivery options". I guess *your* intuition is, if there's only one path, that path must be kept up to date; if there are two, either of them can fail. Or maybe your intuition is based on flooding, which is simple. Although flooding should have low complexity, intermediate choices between single-hop and flooding seem to have *more* complexity than the extremes. For example, maintaining a RON network or a DHT is quite complex, more so than BitTorrent for example. I think your metric seriously underestimates the implementation complexity of DHTs, and allows protocols to "cheat" to lower their complexity. Consider a protocol that wants to transfer a local value from x over a long path with d transport dependencies. The complexity of this is d. If instead of generating one local value at x, the protocol generates m local values, all of which can be used to compute the desired state at the remote node, the complexity of getting the state would go down to d/m. This feels wrong. What if you were to split "any" state derivations into categories? For example, you could distinguish "any of *m carefully chosen* inputs" from "any of *m arbitrary* inputs".

I didn't understand why a wireless node that blindly broadcasts doesn't count for transport dependency; if it fails, the system still fails, right?

Discovering Dependencies for Network Management

—

Knowledge of the inter-dependencies in a distributed system (between hosts, applications, in-network boxes, etc.) would be useful for diagnostics, managements and so forth but is also hard to figure out. The paper proposes to apply machine-learning techniques to measured network traffic to discover this graph of dependencies.

It's nice to see some attempt to model the underlying communication relationships explicitly and causally, with the goal of using it to find problems. I realize that similar systems have been tried with more invasiveness, but this one seems to be happy with just network traffic, so I think that's a good step forward. It would be useful if the authors gave some examples of the kinds of problems that they found, particularly the ones that were more transient, and which would be harder to find with other debugging approaches.

This is a neat idea, and it will be very interesting to see how it works out, but is it possible that a discussion on this topic is most useful when accompanied by comprehensive evaluation and results?

Can you dig deeper in discussing what types of dependencies the proposed approach can infer? For example, it doesn't seem like this could extract dependencies that aren't necessarily linked in time or by protocol: e.g., you download a piece of software from a (bad) server, and that software later initiates a (on-the-wire) seemingly unrelated exchange.

Is it possible that the proposed approach might just make management/diagnostics more complex? Imagine the plight of a sysadmin trying to figure out why his diagnostic system raised an alarm when the alarm is the consequence of a complex machine learning algorithm with seemingly obscure thresholds and parameters all over the place.

I believe Bob Braden has mentioned that Jon Postel wor-

ried that the Internet had gotten (or would get) to the point where it couldn't be rebooted because of circular dependencies.

FLEXLAB: A REALISTIC, CONTROLLED, AND FRIENDLY ENVIRONMENT FOR EVALUATING NETWORKED SYSTEMS

—

Proposes a hybrid emulation/live-deployment scheme for assessing network systems in a controlled-but-realistic fashion. The core idea is to associate with emulated nodes with actual live nodes as counterparts, and incorporate the network conditions experienced by the live node back into the emulation for greater fidelity.

The problem of finding a good platform for network experimentation is an important, and, importantly, open one. Having used (and abused) PlanetLab quite a bit I couldn't agree more that it is overloaded to the point of uselessness. This paper takes a step towards addressing the problem of determining the characteristics of the emulated links: it's hard to argue that link characteristics derived from measurements being made in real time of real links with (near) identical application load won't be accurate. As long as the correct metrics are being measured, that is.

The main benefit of the system proposed in this paper would seem to be psychological for the experimenters, a feeling of "currency", of "running on the real Internet", etc.

Far too much weight is put on the system being "realistic". Emulation cannot in general gather results that are "statistically significant" for the Internet. We simply don't understand how micro-properties like the delays between two adjacent packets affect higher-level measurable properties. You cannot possibly remove "*any* artifacts that might be introduced by special measurement traffic", and trying to emulate all of the "fascinating" details of the Internet seems impossible. For instance, I never would have thought to model a gateway that drops all fragmented IP packets silently. Or to connect a node in China to only a few nodes in the US.

There's a fundamental issue regarding *lag* in sending reports of the network dynamics a live node experiences back to the emulation. Since the live nodes can be deployed at distant locations, this latency will impose fundamental limits on the sort of network dynamics that the emulation environment can incorporate. How serious are those limits? What sort of constraints do they place on the nature of high-fidelity emulation? What classes of studies can the hybrid support, and what classes are beyond it?

Using maximum one-way delay as an approximation of bottleneck queue size seems quite risky. Often, extreme outliers are due to surprising causes/pathologies. And why is it reasonable to assume that congestion mostly happens along the forward direction of a path? Surely this is only more likely if the load induced by the flow itself is likely to tip the scales in terms of congestion.

It's a complicated system. If the goal is to provide nodes with PlanetLab-like link characteristics but lightly-loaded CPUs (à la Emulab), couldn't we just put more machines at the ends of the existing PlanetLab links? If this system were in use and there were 10 times as many Emulab nodes as PlanetLab nodes, wouldn't that be equivalent to having an extra 10 machines at each PlanetLab institution (as long as playback features were still available)? After all, this system doesn't reduce load on PlanetLab links, it only changes CPU utilization.

INTERCONNECTION DISCRIMINATION: A TWO-SIDED MARKETS PERSPECTIVE

—

Describes ISP price setting as a two-sided market: the content provider market on one side and the eyeball market on the other. The authors use this two-sided-market model to show that it is rational behavior for ISPs to subsidize one market at the expense of the other in an attempt to increase their overall profit. The paper also explains through two-sided-market arguments why ISPs are subsidizing residential access at the expense of the (presumably more competitive) content provider market, which has multihoming. Or alternately, it explores how a particular branch of economic theory, Industrial Organization, might inform engineering design for the Internet.

This is an interesting paper and I believe the networking community could benefit from hearing your view of the pricing strategies of ISPs. My main complaint is that you do not clearly explain the assumptions about the cost model for the ISPs. The most intense discussions may be about the assumptions you base your arguments on, not about the internal logic of your argument.

The intent of this paper seems to be to convince the networking research community that economic considerations are as important as technological ones when designing protocols or architectures for the Internet. In this, the paper fails. The claims that this theory should impact Internet research are never justified—the ideas seem more relevant to engineers employed by ISPs—and the paper isn't terribly accessible to the non-expert. It's a current and interesting topic, especially given the ongoing network neutrality debate, but not appropriate material for HotNets.

ACHIEVING GOOD END-TO-END SERVICE USING BILL-PAY

—

Presents a mechanism to provide good end-to-end service between arbitrary endpoints by adding billing information to individual packets. Each ISP is supposed to retain parts of the so-called "nanopayments" associated with a packet: The better its service is, the more "nanodollars" it should receive. The paper argues that such an approach provides better end-to-end service quality, helps to defend against network floods, and discourages spam.

I'm not entirely convinced but it's certainly creative and thought-provoking.

I would think the *granularity* of Bill-Pay, which is per packet, would be wrong, and one would want granularity at the level of an e-mail message. Also, if you did this, isn't there a danger of confusing the email nanopayment with the nanopayment for the transport (TCP/IP)?

I am skeptical that it will fly. Better end-to-end service can probably be bought using such a mechanism, but my concern is that users will be unwilling to use Bill-Pay since it expects them to pay extra money without knowing in advance what kind of performance improvement they are likely to experience. It's only after they have sent some (non-trivial) amount of traffic along a path that they discover what kind of performance they get. This may mean that not many users will use nanopayments, which means that the investment into Bill-Pay infrastructure will go to waste.

The digital cartographer and secretary are central, but is a digital cartographer feasible? It appears that there are just too many paths to keep track of. For each destination of interest, there will be an exponential (in the number of ISPs along the path) number of paths. Furthermore, is a digital secretary feasible? The general problem of detecting malicious activity from normal user behavior is very hard. Even in the limited setting of DoS attacks and spam, people have been talking about it for a while but nothing convincing has materialized. Similarly, your suggestions on preventing man-in-the-middle attacks significantly drives up the cost of deployment.

Some concerns: The question of route stability is a central one but isn't addressed at all here. Didn't people give up on load-based routing because of stability problems? What is the impact of strategic behavior by ISPs? For example, a large ISP that is willing to operate at a loss for some period of time in order to put some smaller competitor out of business? How do you secure the user and ISP OADs from bad receivers and ISPs. What happens if the leftover nanopayment is insufficient to cover the remainder of the path? And the proposed DoS defense seems to imply that attackers can make legitimate clients pay more to access the server. This doesn't seem like a good idea at all. How can you force email senders to include a payment with their messages? How do you introduce the approach? How does it work for UDP? With TCP you can do "piggybacking", but with UDP, is extra signaling required?

Fighting Coordinated Attackers with Cross-Organizational Information Sharing

—

Outlines the design of a system to allow a small number of sophisticated network monitors ("detectives") to make use of observations made by large numbers of other machines ("witnesses"). The network monitors use the observations of witness machines to aid the discovery of bad actors in the network (e.g., a bot net). The query mechanism ensures that private information isn't revealed to witnesses, and that witness replies are believable, via a combination of hashing and encryption.

This paper is well written and describes an interesting vision. The high-level concept sounds great: it's an excellent idea to draw on observations from multiple places taken with "simple and generic traffic monitoring devices", and the scheme for sharing information seems very cunning.

The architecture of the described system is clear, but its potential benefits are only alluded to. Testing the ability of witnesses to aid the detection of bots via control traffic would be a great addition.

Another deterrent for the detectives who would consider "fishing" for private data at the witnesses is post facto auditing. If the logs at the witness show that a detective was engaging in impermissible fishing, that detective might be excluded from the system. As it is probably hard to get in, this would be a serious disincentive. Relying more on this disincentive could allow more query flexibility.

This is nice work that will most certainly move forward the efforts to put together a network-wide defense against many classes of computer hijacking techniques. The biggest problem I have with this paper is that the entire solution was pretty predictable, and the problem statement itself had nothing surprising either.

The paper leaves a lot of questions unanswered. Witnesses "log the facts", but what does this actually entail? How long are records kept for? And with how much detail? If a single witness can reveal "a wealth" of information about which hosts have downloaded the code, then witnesses are expected to be in the network, i.e., routers. If witnesses might be highly resource-constrained then it's even more important to think about the storage and processing costs of being part of this architecture. How do detectives locate witnesses? Does a witness somehow advertise itself? Since witnesses must run the software to answer queries from detectives, there's an upfront commitment to participating, so presumably this information could be stored centrally. Does every detective need to know about every witness? How much coverage of the Internet by witnesses would be required for the system to be effective? Are there timeliness constraints on queries (surely there need to be)? Also are witnesses aware of the identity of detectives? Is the encryption of returned tuples primarily intended to hide the extra records produced by collisions from the detective? or to hide the information from third-party observers? or to verify the witness's statement (in which case a cryptographic MAC could have been used instead of Kerberos-style abuse of encryption)? Maybe it's all three? More explicit mechanisms would help disambiguate this question (e.g. using MAC and encryption would make it clear that both privacy and verification were desired).

So full of holes it will probably generate plenty of discussion.

BLACK BOX ANOMALY DETECTION: IS IT UTOPIAN?

Proposes an anomaly detection framework for network data that separates the anomaly detection module from the data manipulation module. Transformations can be applied to a wide variety of network measurement data until they are brought into the form of real valued constant-spaced time series, which can then be fed to the anomaly detector.

The ability of the proposed framework to detect a very wide range of network anomalies based on very different sources of network data using the same "black box" anomaly detection procedure is a real strength. The paper is well written, timely and shows clear concepts. There are currently no surprising or outraging novelties, but the approach is important for the research community and has a lot of potential. I'd like to see this being used on real data to see where the limitations are and how much can be added if you combine this approach with some domain knowledge. Still the fact that this is in principle domain-knowledge-agnostic is a very strong aspect of this approach.

What kinds of anomalies are most promisingly detectable? We note a recent trend towards stealthy attacks, such as Shrew and RoQ attacks. The discussion should be steered towards the need of new definitions of what "anomalies" are. Flash crowds are anomalies as well, but not malicious.

The framework, as is sadly typical for things that claim to be "frameworks", attempts to encompass any possible implementation. This is a bad property: if everything is allowed, nothing is defined. Much of the framework is pretty obvious as well. This paper is short on specifics.

To what extent do you assume time synchronization? Do you really need "a hierarchy of assumptions"? What would this mean? Sounds complex.

GLAVLIT: PREVENTING EXFILTRATION AT WIRE SPEED

How do you ensure no private/protected data leaves your network? The Glavlit approach centers on a whitelist specifying what content may leave the network. This whitelisting is done out-of-band; a verification box sits at the boundary of the network and checks that data leaving the network is on the whitelist. While fairly straightforward in the abstract, things get tricky when dealing with dynamic content or preventing covert channels. The paper discusses these issues and proposes seemingly workable solutions.

An interesting an perhaps increasingly important problem, but one gets the feeling that the proposed system is going to get very hairy quite quickly (dynamic content, unusual but legitimate protocol use, etc.). Is there something protocol designers could be doing differently to make this an easier problem?

Seems a heavyweight mechanism, and not that interesting.

This paper acknowledges that preventing exfiltration is impossible and actually gives an example of how it can be done while evading Glavlit detection. The system can slow down the information leakage to maybe thousands of times, but probably not millions of times, below the link speed. But the system is geared towards high-speed networks, so even if the spy needs to increase the amount of traffic by a factor of 100,000, it takes only 100 MB to leak a 1000-byte confidential memo in minutes on a fast-Ethernet link (even if there is significant competing traffic). The possibility of the spies discovering more efficient exfiltration channels is also menacing to the proposed system.

I don't understand the motivation behind the solution. You are basically saying that you have an internal network with a mix of public and private content, and you only want the public content to leave the network. Why wouldn't you do something simpler? Such as make sure that the web server contains only public content, and only the web server is allowed to send data outside; or trust the web server itself to deliver only the public content (just like you trust the warden)? You wouldn't need to worry about covert channels, which, as you say, can only be mitigated (not eliminated). Did you consider and reject such simpler options for certain reasons? Could you elaborate on those?

Your discussion of protocol channel mitigation is unclear. A more clear of description of threats, mitigation techniques and their effectiveness would be helpful. Tell us how to think about the threat model.

(R)Evolutionary Bootstrapping of a Global PKI for Securing BGP

Yih-Chun Hu, UIUC
David McGrew, Cisco Systems
Adrian Perrig, CMU / CyLab
Brian Weis, Cisco Systems
Dan Wendlandt, CMU / CyLab

ABSTRACT

Most secure routing proposals require the existence of a global public-key infrastructure (PKI) to bind a public/private key-pair to a prefix, in order to authenticate route originations of that prefix. A major difficulty in secure routing deployment is the mutual dependency between the routing protocol and the establishment of a globally trusted PKI for prefixes and ASes: cryptographic mechanisms used to authenticate BGP Update messages require a PKI, but without a secure routing infrastructure in place, Internet registries and ISPs have little motivation to invest in the development and deployment of this PKI.

This paper proposes a radically different mechanism to resolve this dilemma: an evolutionary Grassroots-PKI that bootstraps by letting any routing entity announce self-signed certificates to claim their address space. Despite the simple optimistic security of this initial stage, we demonstrate how a Grassroots-PKI provides ASes with strong incentives to evolve the infrastructure into a full top-down hierarchical PKI, as proposed in secure routing protocols like S-BGP. Central to the Grassroots-PKI concept is an attack recovery mechanism that by its very nature moves the system closer to a global PKI. This admittedly controversial proposal offers a rapid and incentive-compatible approach to achieving a global routing PKI.

1 INTRODUCTION

The Border Gateway Protocol (BGP) is deployed as the main interdomain routing protocol of the Internet. As described by RFC 1771 all routers in all Autonomous Systems (ASes) are trusted. However, as the Internet has grown, this ubiquitous trust assumption has been proven problematic. For example, in the "AS 7007 incident" one ISP announced short paths to all destinations [10] which caused a wide-spread outage of network connectivity. Clearly, given the importance of the Internet today, we need a more secure routing infrastructure to prevent a single ISP from being able to cause global damage.

Researchers have proposed several protocols to secure BGP [3, 6, 8, 15]. Most of these protocols require that routers authenticate the owner of a network prefix. For example, S-BGP proposes to authenticate prefixes using a PKI that is rooted at IANA [8], as Figure 1 shows.[1] The idea is that IANA is the trusted root of the PKI, and that all participants use IANA's public key to authenticate other certificates regarding prefix ownership. When IANA delegates IP address space to ARIN, it issues a certificate signing ARIN's public key and the IP address space, which indicates that ARIN can rightfully use and delegate those address blocks. Similarly, ARIN will sign AT&T's public key and address space delegation, etc. Some secure routing protocols also need certificates for each AS, which can be achieved through another PKI rooted at IANA, but with certificates binding an AS number to a public key. In this paper, we focus on creating a PKI for verifying route originations, which prevents route hijacks, the most prevalent type of routing attack and misconfiguration on the Internet today. While not discussed in detail, creating a PKI to bind public keys to ASes can benefit from the same "grassroots" approach advocated in this paper.

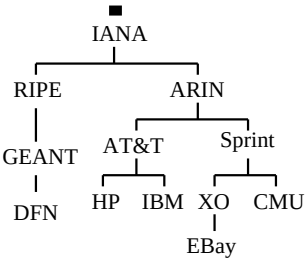


Figure 1: Example prefix PKI structure proposed by S-BGP. IANA is the sole trust root, and each entity in the figure signs the certificate of the entities connected to them from a lower level of the hierarchy. This process mirrors the delegation of IP address space.

S-BGP and soBGP [14] both require a global PKI for AS numbers and prefix ownership in order to provide security guarantees. While S-BGP proposes a PKI with a single root at IANA, soBGP also considers a scenario where a root of trust is formed by large ISPs signing and trusting each other's certificates. The more recent SPV protocol [6] simplifies the PKI requirement slightly by not requiring per-AS certificates, but still needs a global IP address space PKI to authenticate routing announcements.

Unfortunately, setting up such a global PKI is challenging. It requires a significant up-front investment by parties like IANA to manage the private keys, organize outdated and incomplete registries, and issue certificates to ASes. Secondly, all participants need to agree upon and trust a particular root certificate authority (CA); creating a significant point of contention that can stall adoption.

[1] IANA is empowered to allocate address space, but they contract the actual task to ICANN. Thus, while we assume IANA as the logical root of the PKI, this task may well be delegated to another entity.

While these requirements are by no means insurmountable, centralized entities like IANA face little pressure from ISPs to make progress, because no secure routing protocol that requires these certificates has been deployed. This highlights the mutual dependence between the adoption of a new secure routing protocol and the existence of a routing PKI. Stated another way, an AS currently has little demand for an IANA-signed address space certificate, because other ASes do not currently run a routing protocol that chooses routes based on these certificates. Yet operators will not adopt and deploy software for a secure routing protocol if its security benefits depend entirely on a non-existent PKI.

We propose *evolutionary incremental deployment* as a revolutionary approach to bootstrap a secure routing protocol: initially, prefix owners generate and use self-signed certificates, completely without the need for a centralized PKI. As adoption increases, more trusted parties (e.g., tier-1 ISPs) can sign these certificates to resolve any conflicts and provide added robustness for participants, still without requiring the involvement of centralized registries. Finally, driven by a desire to reduce the risk of having distributed points of trust, the system may reach the point where demand for centralized authentication motivates action by actors such as IANA.

In this paper, we study how to overcome this interdependence problem and the lack of incentives for networks to deploy secure routing. We suggest a Grassroots-PKI: an evolutionary approach to deploy a global routing PKI that will enable the deployment of a secure routing protocol. Our goal is to provide a viable deployment path from no security in routing to a highly secure routing infrastructure. We consider the three transitions from no deployment to small deployment, from small deployment to large-scale deployment, and from large-scale deployment to global deployment.

To achieve a viable deployment strategy, we need to provide incentives for ISPs and network administrators to follow each transition. Clearly, the evolutionary approach does not provide as much security as an immediate global deployment of secure routing. However, the evolutionary approach significantly reduces deployment barriers and is strictly better than the absence of routing security. Our approach provides improved security for some networks and worse security for none. If this scheme delayed the adoption of a global secure routing PKI, one could argue that it was detrimental to the greater good. However, quite the opposite is true: the grassroots PKI is specifically designed to hasten the advent of global routing security, by providing powerful incentives to participate in a routing PKI. Specifically, we provide extremely low barriers to joining the PKI, by letting any prefix-owner announce a key. Additionally, we design the deployment path such that when an attacker illegitimately originates a route, the recovery process inevitably moves the routing infrastructure toward a secure global PKI hierarchy. We feel this approach is promising, as it drives a network to be as secure as it needs to be.

2 RELATED WORK

Mechanisms to Authenticate Public Keys:

The most common PKI in use today is managed by corporate CA's like Verisign, who issue public key certificates used by servers for SSL/TLS-enabled protocols like HTTPS. With HTTPS the browser authenticates the server by verifying that the server's public-key is signed by the key of a "trusted root CA". However, due to the large number of online entities that must be verified and cost constraints, CAs can traditionally perform only lightweight identity checks before issuing certificates. In fact, there exists a known case where a hacker obtained a certificate signed for Microsoft [1]. Additionally, because of a focus on usability over security, current web browsers contain root key certificates from over 30 different CA's. Having a large root of trust weakens the security of the overall system, because an attacker that compromises a single CA can forge any web site. This demonstrates that while having many different trust roots eases usability and adoption, it lacks the strong security desirable in a full routing PKI.

More flexible and inexpensive mechanisms for establishing trust without a centralized authority also exist today. The web of trust in pretty-good privacy (PGP) authenticate public keys based on a graph of mutual trust relationships [16]. Unfortunately, the security of such trust paths quickly deteriorates even for extremely small numbers of links [12]. Alternately, the SSH protocol supports a "leap-of-faith" authentication model, in which users accept an unauthenticated key upon first connecting to a server, and use this key to verify all subsequent connections. While it offers no security for the first connection, further communication enjoys significantly improved security and the simplicity of this model is widely recognized as a reason SSH saw quick and widespread adoption.

A grassroots PKI will require the ability to merge separate smaller PKIs into a single larger PKI. One of the largest efforts to build a PKI with many administrative entities was the Automotive Network Exchange (ANX) [11]. A central goal of ANX was to bridge trust between the PKIs of the member sites. For the member sites to communicate securely, various ISPs also needed to participate in the PKI. For various reasons, ANX did not fully deploy. One of these reasons seems to be the difficulty in setting up the trust between all of the members simultaneously.

Finally, similar to BGP, securing DNS exhibits a dependency on PKI deployment, because DNSSEC requires a hierarchical PKI mirroring domain name delegation in order to authenticate DNS records. Top-level domains (TLDs) like .com need to publish public keys and sign certificates for sub-domains before that sub-domain can

provide secure DNS responses. To circumvent this dependency, a recent proposal called DNSSEC Lookaside Validation (DLV) [13] permits domain keys to be signed by non-TLD "trust anchors" prior to the existence of a full PKI.

PKIs for Secure Routing:

S-BGP proposes a single PKI root at IANA and a structure that mirrors address delegation. It allows for incremental deployment, but accepts a path as "secure" only if the prefix ownership and AS-path can be completely verified. This requires each AS in the AS-path to have an IANA-rooted certificate before a particular announcement is considered secure. Therefore, S-BGP does not allow for incremental deployment of the authentication infrastructure.

The soBGP effort proposes a PKI that is incremental by nature, where a PKI is generated based on which entities participate and whom the participants choose to trust. However, it recommends no particular structure for that PKI, nor does it provide a design specifically aimed at incentivizing participation in the PKI.

The SPV protocol suggests leveraging identity-based cryptography (IBC) [2] to simplify certificate distribution. SPV uses the prefix as a public key, requiring the prefix owner to contact a root CA to obtain the corresponding private key.[2] However, before any routing information can be authenticated, SPV still requires that all participants trust the global CA, that the CA can identify the legitimate owners of each prefix, and that all participants possess the CA's public key.

3 A Step-wise Approach for Bootstrapping a Routing PKI

Establishing a large PKI for the 20,000+ organizations involved in BGP routing is a daunting challenge, even when compared to initiatives like ANX (mentioned in Section 2) which have struggled with deployment. Moreover, the heterogeneity of entities in the Internet is significant, as ISPs span continents, languages, political ideologies, and cultures and no single entity can mandate a solution. These impediments suggest that the establishment of a PKI will not occur overnight and that individual actors must have strong economic incentives to overcome these barriers to participation. An evolutionary approach to building a global PKI can minimize these hurdles while still achieving strong security as an end result.

In this section, we present a multi-phased, evolutionary approach for establishing a global PKI. We start out assuming an Internet with mutually distrusting entities, with the goal of achieving a global PKI that enables any participant to authenticate any prefix.[3] We suggest two intermediate steps en-route to a full PKI: first, independent simple PKIs based on self-signed certificates; and second, small hierarchies of independent complex PKIs that certify their customers. For each case, we discuss how to reduce the associated security risks while simultaneously providing incentives for adoption. At each step, ASes have strong economic motivation to participate and any successful attacks will automatically drive the infrastructure toward a global PKI. Thus, our evolutionary approach begins with scattered trust points, and culminates in a global PKI with universal trust. Though we share the same final goal as previous routing PKI proposals, this bottom-up approach can greatly accelerate the process.

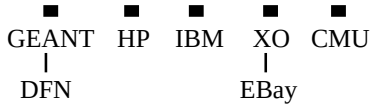


Figure 2: The flat distribution of trust with self-signed certificates. The top five entities are trust roots with self-signed certs, while DFN and EBay has certificates signed by their ISP's self-signed certificate.

3.1 Self-signed Prefix Certificates

The administrative entities controlling authorization (i.e., the Regional Internet Registries (RIRs) or large ISPs) may or may not initially participate in a secure routing PKI. Even if they do, the chain of trust extending from these entities may not follow the existing address authorization infrastructure, because it is likely that some ASes lower in the hierarchy will want to adopt even though entities above them in the delegation chain are not yet participating. Therefore, the PKI for a secure BGP routing infrastructure should be prepared to begin simply, for example, by dealing with several trust roots [9].

We propose a grassroots-PKI, where *anyone* can start disseminating a self-signed certificate for a prefix, drastically lowering the complexity and cost of participation. With no verification process required to claim a prefix, this revolutionary approach has seemingly severe security failures. However, this initially loose structure can rapidly transition to a high-security global PKI because any attack makes the network *more* secure as a result—thus, malicious actors are placed in a quandary where *the best attack strategy may actually be to not attack at all*! Furthermore, as we outline below, simple rules can assure that new vulnerabilities are not introduced into the routing system during this incremental process.

In this stage, an AS may unilaterally decide to sign and announce a key for each of its prefixes without any external coordination, or any ISP may use its own self-signed certificate to delegate prefix ownership to customers. Figure 2 shows an example of small independent trust realms using self-signed certificates.

ASes must simply disseminate these prefix public-key certificates and use the corresponding private key to sign prefixes in routing announcements. *Each key-pair is*

[2] Some people believe that identity-based cryptography obviates the need for a trusted CA, which is unfortunately not the case.

[3] As discussed earlier, a similar "grassroots" concept could also help the adoption of a global AS PKI, if required by the routing protocol.

bound to a single prefix, and that key can only authenticate routing updates associated with that prefix or its sub-prefixes.

Because they are self-signed, these certificates do not imply an endorsement from a centralized authority like IANA that the AS originating the prefix is its legitimate owner. However, these self-signed prefix certificates can be used to authenticate address space delegation between parties within the grassroots-PKI. For example, if a large ISP has a trusted prefix key, it can sign the key of any customer announcing a smaller portion of that address space, indicating that the ISP permits the customer to announce that prefix.

Self-signed certificates are distributed as transitive attributes within the BGP update message, meaning they will be forwarded with route announcements even by ASes that are not yet participating in the secure routing scheme.

The assumption made here is that announcing a self-signed certificate provides security benefits for the early adopters, because BGP routers apply the following list of precedence to decide which BGP prefix/key pairs to trust:

1. **Root-signed:** Prefix that is secured by a certificate chain rooted at IANA.

2. **Trust-anchor-signed:** Prefix with a certificate-chain rooted at a well-respected "trust-anchor", such as a tier-1 ISP, registry, or corporate CA. Such an oligarchy of trusted entities is similar to current web security, where browsers ship with a relatively large list of trusted certificates.

3. **Self-signed:** Prefix signed by a key not associated with a trust anchor. For multiple such certificates, the oldest certificate (date first seen by the router, not date carried in certificate), is preferred. This model is similar to light-weight destination authentication in SSH.

4. **Unsigned:** A prefix in a BGP update as announced on the Internet today.

Note that a BGP router has the highest preference for prefixes certified through trusted entities, which can "overrule" other certificates for the same prefix that are only signed by less well-known entities. The key used by a trust anchor to sign prefix certificates is not itself a prefix key, meaning that a trust anchor can sign prefix keys even if it does not own the associated address space. This flexibility enables quick PKI development despite organizations in the delegation hierarchy that do not yet participate in the PKI. Routers install a trust anchor's public key (used to verify prefix certificates) only if it decides that party is indeed trustworthy.

The policy of preferring older self-signed certificates not only protects the address space of participants from an attacker's unauthenticated route announcement, but it also encourages early adoption because creating a self-signed certificate early (i.e., before an attack) is much easier than later demonstrating prefix ownership to a trust anchor in order to reclaim. By adopting early, an AS achieves a high level of security (an attacker must deceive a trust anchor to be successful) at an extremely low cost.

Accepted certificates/prefix pairs are placed in a local database along with a timestamp indicating when the prefix was first seen at that router. New routers just coming online can be easily be pre-configured with certificates learned by other routers to immediately begin choosing secure routes.

Risk. This approach has two main risks: first, an attacker may use self-signed certificates to try and divert traffic from a legitimate prefix owner, and second, an attacker may compromise one of the trust anchors and issue illegitimate certificates. We explore both possibilities.

Risk 1: Preferring older self-signed certificates prevents an attacker from stealing a prefix that has already been self-signed by its owner. However, an attacker could announce a self-signed certificate before the legitimate owner. Our goal in this case is two-fold: first, make this attack difficult, so that malicious actors do not gain any attack power with a grassroots PKI compared to BGP today. Second, provide a straight-forward mechanism to resolve this conflict that results in an even more secure infrastructure.

To provide the first property of introducing no additional vulnerabilities into the system, a router only accepts a self-signed prefix key if that key has been propagated with every preferred route to that prefix for a set period of time (e.g. 24 hrs.). This simple yet effective heuristic is similar in motivation to PGBGP [7], and builds on the intuition that at any point of time, most Internet routes are correct. Invalid originations for actively-used address space result in outages, which even today are recognized and manually filtered on human timescales of several hours at the most. With this rule, malicious key announcements cannot violate existing security mechanisms like filter lists or make it easier for an attacker to divert traffic. Thus all ASes, even those not participating in a Grassroots-PKI, are no more vulnerable to attacks than they are today.

If an attacker nonetheless successfully has its route and key accepted, we rely on the policy of preferring certificates with a higher trust level as a mechanism for "revoking" the invalid ownership claim. For example, if ISP evil.net is first to issue a self-signed certificate for one of angel.com's prefixes, angel.com can regain control by getting a trust anchor (for example, a tier-1 ISP responsible for providing their transit connectivity) to sign angel.com's prefix key. This makes angel.com's key more trusted than the key from evil.net, and angel.com will quickly reclaim its address space. The required chain of communication largely mirrors today's use of reac-

tionary BGP filters to block invalid routing announcements. However, the major difference is that with a grassroots PKI, the destination now become significantly more resistant to all future attacks, and the overall routing system is one step closer to a global PKI.

A related concern is an attacker's ability to announce a new unsigned sub-prefix of another prefix that is already signed[4]. Without a top-down PKI it is difficult to determine whether this sub-prefix is a valid route from a network not yet participating in the grassroots-PKI, or an attack meant to illegitimately divert traffic. The scheme must either accept and use less-trusted sub-prefixes, introducing significant vulnerability into the system, or reject all more specific prefixes unless they are signed by a key as or more trusted than the prefix they deaggregate. We choose the later, because legitimate sub-prefixes in global routing tables are likely to be IP space obtained by multi-homed customer from one of its upstream ISPs. As a result, sending traffic to the larger prefix will still result in the data being correctly delivered to the sub-prefix owner. If the sub-prefix owner wants its sub-prefix accepted globally as a secure route, it can easily have its upstream delegate that address space by signing the customer's key for the sub-prefix.

Risk 2: While self-signed certificates provide protection against common BGP attacks and misconfiguration, the large number of trust anchors still represents a legitimate vulnerability. Because any trust anchor certificate is preferred over all self-signed certificates, a prefix with only a self-signed certificate is vulnerable to the compromise of any trust anchor. Yet this preference of trust anchors over self-signed certificates is required as part of the attack resolution process described above. Thus, as demand for security increases, destinations will logically desire to have their self-signed certificates be signed by a trust anchor, even if no attack has yet occurred. This leads to our next stage of adoption: independent complex PKIs.

3.2 Independent Complex PKIs

For added robustness, we consider an architecture where islands of domains have their originally self-signed keys certified by one or more entities designated as "trust anchors", thus beginning to form a PKI hierarchy.

As mentioned above, the resolution of routing attacks creates a certification chain from a trust anchor to the legitimate prefix owner. Additionally, security conscious prefix owners are likely to preemptively have their prefix keys signed by trust anchors to gain improved attack robustness. ISPs will also gain a competitive advantage if they offer customers a certificate path to a trust anchor. In the course of this process, the trust anchors essentially become the roots of smaller hierarchical PKIs. Figure 3 shows an example, where the formerly self-signed clusters from Figure 2 are collected and authenticated by three different trust anchors.

[4]The announcement of a super-prefix is not a security concern, because IP forwarding will prefer the more specific valid route

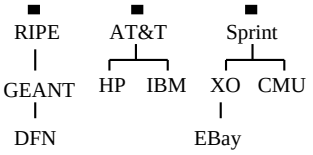


Figure 3: Three independent complex PKIs, each with a trust anchor at its root. Trust hierarchy does not necessarily mirror address delegation.

Risk. This approach has two main risks. First, operational confusion may occur during the transition from the primary use of self-signed certificates to independent complex PKIs. Who is qualified to be a trust anchor? Who decides if a trust anchor should be removed because of bad security practices? Similar to the inclusion of trusted keys in a browser, community consensus will play a powerful role in handling such issues. ISP operational organizations (e.g., NANOG) will be able to develop policies, likely placing trust in organizations already allocated significant responsibility for running core network infrastructures.

Second, the many trust anchors at the root of independent PKIs are still a vulnerability, as compared to full-time CA's, these organizations likely spend less money on and have less experience with roles like validating the identity of a prefix owner and protecting the private signing key from compromise. Prefix owners can achieve additional robustness either by having their key signed by multiple trust anchors or by the most trusted of entities, IANA. Either option is a viable path toward reaching the third and final stage: global PKI.

3.3 Global PKI

With the existence of many independent complex PKIs, we have clearly overcome the mutual dependence cited earlier as a key stumbling block to deployment of a full routing PKI. The existence of a number of trust anchors will provide an incentive for the establishment of a smaller root of trust. Each trust anchor can offload a considerable administrative burden onto the new trust root, and at the same time reduce its security exposure. This economic incentive is important, since any entity assuming the burden of acting as a trust root brings upon themselves a considerable liability. We believe either IANA or a small number of the most well-respected trust anchors will fill this role. There are two likely scenarios for a global PKI: cross-certification or consolidation under a single-rooted hierarchy.

3.3.1 Cross-certification

Large ISPs at the root of independent complex PKIs may be willing to cross-certify each other on the basis of existing business relationships. But in the eyes of some, direct cross-certification "turns the hierarchy of trust into

the spaghetti of doubt, with multiple certificate paths possible from leaf to roots ..." [4]. With cross-certification any given BGP participant may find it difficult to known where trust is coming from, or how reliable that trust is.

An alternative to direct cross-certification is the use of a Bridge Certification Authority (BCA) [5]. A BCA is a CA trusted by all of the smaller PKI roots to mediate trust between them. Each PKI root cross-certifies once with the BCA, and trusts that the BCA will correctly mediate policy and trust the various roots. Any mutually trusted entity could become the BCA in a secure BGP, but IANA may be the most natural choice. Note that as a bridge IANA would not actually require a PKI under it.

Risk. ANX used the Bridge CA architecture, and experienced organizational difficulties due to the number of administrative entities. Similar political complexities may render a BCA infeasible for secure BGP.

3.3.2 Single-Rooted Hierarchy

If IANA and the RIRs agree to participate in a routing PKI, then ISPs and other trust anchors may be willing to graft their root into a Single Rooted Hierarchy [5]. Much like the BCA case, the existence of independent trust anchors creates both management and security incentives to move toward a single root. Additionally, once certificates become a key part of the routing protocol, centralized address space delegators like IANA will be more willing to participate because they could gain power over wayward address owners by denying them a new certificate.

Risk. Single-rooted hierarchies have difficulties if the root key needs to be revoked. The approach of a single-rooted hierarchy for a secure BGP has the remote, yet real, risk that route authorizations for the entire Internet become invalid, causing a breakdown of interdomain routing because no secure routes can be found.

4 CONCLUSION

The deployment of a global PKI needed for secure routing is not sufficiently incentivized to overcome operational barriers to development and adoption. Contrary to current top-down PKI proposals, we suggest a grassroots PKI, representing a more realistic deployment path that will facilitate development of a global routing PKI and the deployment of secure routing. By accepting an imperfect level of security, but creating incentives for improved robustness, we construct a global PKI through incrementally staged deployment. At no point do we introduce new vulnerabilities, and attacks against legacy security weaknesses result in a strictly more secure network that is closer to our goal of a global PKI. We anticipate that our (r)evolutionary PKI deployment mechanism will encourage a dialog in the secure routing community to consider alternative PKI deployment strategies.

5 ACKNOWLEDGMENTS

This research was supported in part by CyLab at Carnegie Mellon under grant DAAD19-02-1-0389 from the Army Research Office, and grant CT-0433540 from the National Science Foundation, and by a gift from Cisco.

We would like to thank Jennifer Rexford for interesting discussions and feedback, and the anonymous reviewers for their insightful suggestions.

REFERENCES

[1] MS01-017: Erroneous VeriSign-Issued Digital Certificates Pose Spoofing Hazard. `http://support.microsoft.com/kb/293818`.

[2] D. Boneh and M. Franklin. Identity-based encryption from the Weil Pairing. In *Advances in Cryptology — CRYPTO '2001*, pages 213–229, 2001.

[3] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. McDaniel, and A. Rubin. Working around BGP: An incremental approach to improving security and accuracy in interdomain routing. In *Proceedings of NDSS 2003*, February 2003.

[4] P. Gutmann. PKI: It's not dead, just resting. *Computer*, 35(8):41–49, August 2002.

[5] P. Hesse and D. Lemire. Managing interoperability in non-hierarchical public key infrastructure. In *Proceedings of Network and Distributed System Security Symposium, 2002*, February 2002.

[6] Yih-Chun Hu, Adrian Perrig, and Marvin Sirbu. SPV: Secure path vector routing for securing BGP. In *Proceedings of ACM SIGCOMM 2004*, September 2004.

[7] Josh Karlin, Stephanie Forrest, and Jennifer Rexford. Pretty good BGP: Improving BGP by cautiously adopting routes. In *Proc. International Conference on Network Protocols*, November 2006.

[8] S. Kent, C. Lynn, J. Mikkelson, and K. Seo. Secure border gateway protocol (S-BGP) — real world performance and deployment issues. In *Proceedings of NDSS 2000*, pages 103–116, February 2000.

[9] John Linn. Trust models and management in public-key infrastructures. Available at `http://citeseer.ist.psu.edu/linn00trust.html`.

[10] S. A. Misel. Wow, AS7007! NANOG mail archives, `http://www.merit.edu/mail.archives/nanog/1997-04/msg00340.html`, 1997.

[11] Robert Moskowitz. History of the bca concept and other bca efforts, April 2000. Available at `http://csrc.nist.gov/pki/twg/Archive/y2000/presentations/twg-00-15.pdf`.

[12] Michael K. Reiter and Stuart G. Stubblebine. Authentication metric analysis and design. *ACM Transactions on Information and System Security*, 2(2):138–158, 1999.

[13] S. Weiler. Dnssec lookaside validation (dlv), draft-weiler-dnssec-dlv-01.txt. Technical report, IETF, June 2006.

[14] B. Weis, ed. Secure origin BGP (soBGP) certificates. Internet-Draft, July 2004. Work in progress. Available at `http://www.watersprings.org/pub/id/draft-weis-sobgp-certificates-02.txt`.

[15] R. White. Deployment considerations for secure origin BGP (soBGP), draft-white-sobgp-bgp-deployment-01.txt. Draft, IETF, June 2003.

[16] Philip R. Zimmermann. *The Official PGP User's Guide*. MIT Press, Cambridge, MA, USA, 1995. ISBN 0-262-74017-6.

Don't Secure Routing Protocols, Secure Data Delivery

Dan Wendlandt, Carnegie Mellon
Ioannis Avramopoulos, Princeton
David G. Andersen, Carnegie Mellon
Jennifer Rexford, Princeton

1 Introduction

Internet routing and forwarding are vulnerable to attacks and misconfigurations that compromise secure communications between end systems. With networks facing external attempts to compromise their routers [3] and insiders able to commandeer infrastructure, subversion of Internet communication is an ever more serious threat.

Much prior work has proposed to improve communication security with secure interdomain routing protocols (e.g., S-BGP [10] and so-BGP [12]). We argue that solving the problem of secure routing is both harder and less effective than directly solving the core problems needed to communicate securely: end-to-end confidentiality, integrity, and availability. Secure routing protocols focus on providing *origin authentication* and *path validity*, identified as necessary by the IETF to secure BGP [7]. Unfortunately, these properties are both too little and too much:

Secure routing is too little: As we discuss further in §2, secure routing does not completely address the core problems in secure communication. For example, it cannot prevent adversaries on the communication path from eavesdropping or modifying data traffic. Hosts must still use end-to-end cryptography to defend against these attacks. Similarly, secure routing cannot detect or prevent packet loss due to data-plane bugs, misconfigurations, or attacks.

Secure routing is too much: The mechanisms behind secure routing, both cryptographic and administrative, are painfully heavy-weight. They require router hardware upgrades for cryptographic processing, time-consuming maintenance of address registries, and a new public key infrastructure (PKI).

Recognizing that a secure version of BGP will be difficult to deploy, yet provide only limited protection, we ask: what is the best division of labor between end systems (end hosts, or edge routers acting on behalf of end hosts) and the routing infrastructure to provide secure, robust communication? The answer, we argue, is that the routing infrastructure must only provide *availability*, i.e., enable an end system to find a working path to the valid destination as long as such a path exists. End systems can provide confidentiality and integrity as needed.

Following this model, we present Availability Centric Routing (ACR), which is based on three principles:

1. End systems learn multiple paths to a destination.
2. End systems monitor end-to-end integrity and path performance to determine if a path is working.
3. End systems can change paths to find one that works.

By propagating multiple paths per destination instead of one "best path," ACR thwarts an adversary's attempt to prevent a source from hearing a valid path to a destination. Taken together, ACR has several interesting advantages over traditional secure routing schemes:

- Using alternate paths can circumvent data-plane availability threats, such as malicious drops, misconfigured ACLs, link DoS, and transient routing issues.
- Significant gains in resilience are achieved even if only a few interested domains cooperate.
- Adoption is simplified because no address registry, AS-level PKI, or router cryptography is required.
- Performance, usually at odds with security, also benefits from path diversity.

ACR achieves robustness by treating learned routes as possibilities, not certainties. With this approach, control-plane security (e.g., S-BGP) is an *optimization* to help ACR find valid paths quickly by avoiding spurious routes, rather than a requirement for communication security.

2 Threat Model

Reliable Internet communication can be impaired by attackers who compromise routers or by link DoS, failures, bugs, and misconfigurations. In a traditional threat model, attackers can tamper with data or impersonate identities (violate integrity), snoop on traffic (violate confidentiality), or deny service (reduce availability). In this section, we first examine why only the last of these threats—availability—requires support from the routing infrastructure. We then examine in more detail the ways an attacker might attempt to deny availability.

Integrity can be provided end-to-end using well-known cryptographic techniques (Message Authentication Codes) along with shared secret or public key authentication schemes. Data **confidentiality** is similarly easy to protect using encryption. This leaves **availability** as the remaining threat. Unfortunately, cryptography cannot get packets across a path that drops or misdirects all traffic.

Control of a router, legitimate or illegitimate, grants

significant power to compromise communication security in both the control and data planes.

Control Plane: An attacker can influence the *global* flow of traffic by falsifying BGP routing information. By announcing a victim's IP prefix or manipulating the AS path, an adversary can draw traffic to its own routers, where it can observe, modify, or drop data and impersonate the destination. An attacker can also prevent a portion of the Internet from hearing the valid route announcement, "black-holing" traffic to the victim. We term the use BGP route announcements to maliciously attract traffic a "control-plane" attack. Secure BGP proposals impede, but do not prevent, attackers from mounting such attacks by providing *origin authentication* and *path validity*. [1]

Data Plane: Despite reducing an attacker's ability to attract traffic, a secure control plane cannot prevent malicious routers or insiders that manage to be on a legitimate communication path from observing, modifying, or misdirecting traffic. Nor does control-plane security protect against link DoS, or misconfigured packet filters. We term these threats "data-plane" attacks. Data plane attacks are particularly troublesome because BGP (secure or not) will not switch away from a "best path" even if it becomes effectively useless for a particular application.

Because control-plane security must still be augmented with end-to-end techniques to guarantee integrity and confidentiality, we argue that *the only property that the control plane must provide is availability*; that is, it must guarantee that a sender will hear about a valid path to the destination if one exists. The control plane *may* provide information regarding what AS paths are likely to be legitimate, but this information is not a requirement for communication security.

A more subtle threat to confidentiality is traffic analysis, which gleans information simply by observing the pattern of communication between hosts even when data is encrypted. Fortunately, traffic analysis is more difficult than simply black-holing traffic, because it requires that the attacker not only be able to intercept traffic, but also to re-inject it to the correct destination. We suspect, but leave for future work, that the use of path selection heuristics as described in §3.4 will make traffic analysis difficult for all but the most well-connected ISPs. In the case of either ACR or a secure BGP, senders in need of strong protection against traffic analysis are best served by techniques like mixnets[6].

A final threat comes from attackers who advertise unallocated or unused address space, as is sometimes done by spammers to avoid IP address blacklists [14]. We do not consider preventing these announcements to be a central requirement for robust routing, because they do not undermine communication security and are only weakly related to the fundamental economic incentives that fuel the spam problem.

[1] For example, secure BGP cannot prevent announcements that attract traffic by violating BGP policy, such as a customer redistributing routes heard from one provider to another.

3 Availability Centric Routing

The goal of *availability-centric routing* is to enable end systems to communicate securely even if portions of the network infrastructure are controlled by an adversary. ACR uses four components. First, one or more transit ASes act as *availability providers* (APs) that provide the edge with multiple routes for each destination. Second, sources using ACR cryptographically verify the identity of the destination host or network, to confirm that the chosen route reaches the correct destination. Third, ACR end systems securely monitor communication performance; if performance is too poor, for whatever reason (a situation-specific definition), they signal ACR to use a different path. Fourth, the ACR end systems distribute traffic over one or more paths supplied by the AP by applying selection algorithms that quickly identify working paths with high probability.

3.1 Multipath via Availability Providers

To provide path choice in a legacy, single-path BGP environment, ACR includes mechanisms to advertise multiple paths for a single destination and then direct traffic onto these alternate paths. This approach is akin to proposed multipath schemes like MIRO [18]. Availability providers give the network edge access to multiple paths via a (presumably paid) AS-level *deflection service*. End systems can avoid failures by redirecting traffic to different paths.

An availability provider maintains a *route repository* containing all routes learned from BGP peering sessions with neighboring ASes. The repository may be populated by passive BGP sniffers at peering links, or by a BGP monitoring protocol. Customers can request routes on demand from their AP (e.g., if their current path is not working), or subscribe to a feed of paths to particular destinations using either a custom protocol (future work) or the proposed *add-paths* extension to BGP [17].

Sources use alternate paths by tunneling packets using IP encapsulation (e.g., L2TPv3 [11]) to *deflection points* in the AP's network. Paths from the route repository include the deflection point IP address, the encapsulation method to use, and a *deflection forwarding identifier*. This tunneling can be performed at line rate by high-end routers [8] and enables decapsulated packets to circumvent normal BGP routing using directed forwarding. Directed Forwarding uses an alternate forwarding table to route packets based on the deflection forwarding identifier included in the encapsulation header. After decapsulation and directed forwarding, subsequent routers forward the packet normally. Access to the deflection ser-

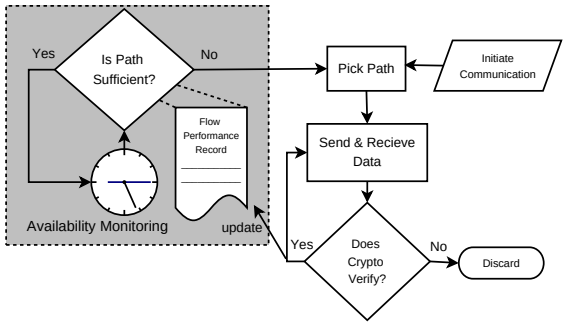


Figure 1: Control-flow of "availability monitoring" in ACR.

vice can be efficiently controlled by light-weight authentication "cookies" such as those found in L2TPv3.

3.2 End-to-End Integrity Check

To work, a path must connect the source to the *correct* destination. ACR allows end systems to authenticate destinations in whatever way they choose, from generic mechanisms such as IPsec or SSL to application-specific approaches like DNSSEC.[2] Many important protocols, including HTTP, SMTP, SSH, and SIP, already support both client and server authentication, and we argue that the majority of important Internet communication *already occurs over secure channels like SSL or IPsec*. Importantly, ACR does not require that all hosts and/or routers participate in a PKI. For example, with HTTPS, clients commonly present no authentication credentials to the server at all, and instead dynamically establish a secret used to verify the integrity of all further packets.

3.3 Availability Monitoring

Detecting availability attacks requires the ability to monitor a network flow and determine if the current path is a usable route.

In the context of Figure 1, consider a general-purpose availability monitor within the TCP stack of an end host using IPSec for end-to-end security. A call to *connect()* causes the path-selection component to select an initial route. TCP sends a SYN packet and sets its retransmission timer. If the timer expires before the SYN/ACK comes back, the monitor records the event and *may* change to an alternate path before retransmitting. Similar monitoring occurs for all data transfered. With TCP, the "flow performance record" consists primarily of state the protocol already keeps to manage reliable delivery, but could be augmented with retransmission or timeout counters to track recent path performance. This record must be reset each time a new path is selected, but no TCP-specific behavior or state is modified. Received packets are verified for integrity using IPsec and are discarded if the check fails, so that paths with adversaries manipulating packets will cause time-outs that result in a path switch.

While this example monitor is simple and general, ACR can work with any type of availability monitoring the edge chooses to employ. In particular, edge routers could use monitoring schemes similar in spirit to Listen [16] or Stealth Probing [4] to detect and switch away from bad paths *on behalf of clients*. Alternately, applications like VOIP clients that already incorporate protocol-specific monitoring could use this information to signal a desire for a different path.

3.4 Path Selection Algorithms

Path selection algorithms should quickly locate working routes, to minimize the time to recover from failures or attacks. These algorithms are triggered by the availability monitors when failures are detected (Figure 1). Path selection algorithms can combine topological information (e.g., AS-paths from insecure BGP) with external knowledge (e.g., known AS connectivity or history of good routes) to select candidate paths. ACR treats this information as *hints*, not truth, because the information may be stale or inaccurate depending on its source. Path selection could explore several paths in parallel to further reduce recovery time at the expense of additional bandwidth. Selection can be assisted by heuristics such as:

Static destination connectivity hints: Destinations that care about availability are likely to know their upstream connectivity. ACR can use this knowledge to give the edge "hints" to quickly identify promising paths. BGP paths that are inconsistent with the connectivity hint from the destination receive lower priority in the path exploration process. Because their consistency is not critical (they affect only priority) static hints can be distributed ahead of time, out-of-band, or via replicated repositories.

Route stability heuristics: Many Internet routes, particularly those to popular destinations, are quite stable [15]. ACR could take advantage historical route information to identify good paths more quickly. Unlike schemes that discard routes that fail historical tests, and so require exceptionally low "false-positive" rates, ACR will still use "anomalous" routes if (and only if) they work correctly end-to-end.

Path ranking and selection can be handled by an end host, an edge router, or even the AP to simplify the functionality at the edge network.

4 ACR WITH LIMITED DEPLOYMENT

In the long term, we envision ACR being used with a globally deployed multipath protocol like MIRO[18]. Yet we demonstrate in §5 that deployment by even a single tier-1 ISP provides customer ASes significant availability improvements in the face of routing attacks.

However, "legacy providers" still running single-path BGP complicate the limited deployment scenario. For

[2] Note that because encryption is not required for integrity, it is needed only if the application requires confidentiality.

example, if a destination D has only a single (legacy) provider P, and P believes and propagates a false route for D, no availability provider would be able to reach D. Therefore, ACR, when deployed at limited locations, requires additional light-weight control-plane countermeasures (simple BGP filters, see §5) to prevent such control-plane availability attacks. Before evaluating the resilience of limited ACR deployment we cover two issues related to using ACR in a legacy environment.

Resisting sub-prefix hijacks: With BGP, an attacker can announce a sub-prefix more specific than a legitimate advertisement. This attack is highly effective because the sub-prefix propagates to all ASes and all routers will forward traffic to the more specific sub-prefix. In ACR, if a destination D is not directly connected to its AP, packets sent by the AP to D via a legacy provider P may be misdirected to an attacker if P believes the attacker's sub-prefix.

To counter this attack, a sequence of legacy providers between D and the AP must not believe the attacker's sub-prefix. ACR ensures this by emulating "flat addressing" using /24's, which is the longest prefix most ISPs will accept (i.e., it cannot be sub-prefix hijacked). In the example above, D can announce its prefixes as /24's to P, so that P will not divert packets. P can safely aggregate the /24's before announcing them to peers or customers, and must announce the longer-prefixes only to one upstream provider. This chain terminates at a tier-1 provider, who is directly connected to other AP's and thus assures that there is a complete path from any AP to D that cannot be sub-prefix hijacked. Effectively, upstream providers accept a moderate increase in routing table size to increase availability for their customers, while the global routing table size remains unaffected.[3]

CIDR addressing, the root cause of sub-prefix hijacks, is also troublesome for other proposals for secure routing. For example, sub-prefixes in forwarding tables can lead to discrepancies between control and forwarding plane paths, lessening the benefit of a verified BGP AS-Path. Similarly, prefix aggregation significantly complicates origin authentication. While we propose an incremental measure for dealing with CIDR above, ultimately we feel that a more sound architectural choice is to move toward a flat addressing model for the Internet.

Resisting deflection point hijacks: A BGP hijack could also block a subscriber from reaching its AP's deflection points if the subscriber's direct upstream provider did not support ACR.[4] Fortunately, the number of deflection point prefixes would be quite small, and they are found within stably connected core networks. These properties facilitate "defensive filters" that explicitly deny route announcements for special destinations on all but a few peering sessions.

5 EVALUATION

We explore the effectiveness of ACR and its countermeasures in the context of today's Internet. In our evaluation, each path may contain at most one deflection point and only a few ASes offer deflections. Our experiments examine ACR's performance against an attacker who announces an IP prefix that belongs to a victim network.

Method: We run simulations on an AS-level graph based on July 2006 RouteViews data with AS relationships inferred using Gao's algorithm [9]. The route selection policy prefers customer-learned routes over peer-learned routes, and prefers provider-learned routes the least, with ties broken using AS-Path length. Each trial has one legitimate AS and a set of attacking ASes that all announce the same prefix. We vary the number of malicious ASes, performing 100 trials for each configuration.

Result 1: A single tier-1 availability provider significantly increases routing robustness compared to stubs using either single-path BGP or intelligent multi-homing. Figure 2 charts the average reachability of the legitimate destinations versus the number of attacking ASes. The bottom line (Single-Path BGP) shows the average success rate of all stub ASes in reaching the destination using normal BGP. We simulate intelligent multihoming by testing all stub ASes with exactly five providers to see if any of their five BGP-learned routes are valid.[5] The availability providers for the Tier-1 AP data include all ten ISPs commonly thought to not purchase transit from another ISP, and makes the reasonable assumption that these ISPs offer deflections on all BGP-learned paths. The results indicate the average success rate for these any end system that is able to use just one of the tier-1 APs.

While intelligent multihoming sources can select from multiple paths, *only a tier-1 availability provider exposing multiple BGP-learned paths to the same destination provides strong resilience to hijacks*. ACR works so well because topology and the common BGP policy of preferring customer-learned routes forces an attacker to be "local" (a customer of all of a destination's providers) to prevent the AP from hearing a legitimate announcement.

Result 2: ACR's availability benefits can be further improved using easily-deployed BGP filtering local to the victim. As shown in Figure 2, adversaries are sometimes assigned to local ASes, reducing the Tier-1 AP success rate to 95% with many attackers (e.g., second from

[3] We have heard from operators that announcing smaller subnets into the global routing table to resist sub-prefix attacks is not uncommon today. ACR offers similar protection but without polluting global tables.

[4] This customer would have an incentive to switch to an ACR-speaking ISP, but we also believe that customers can benefit from using a "remote" (i.e., non-first-hop) availability provider (§6).

[5] A selection intended to capture stubs that have invested significantly in network availability.

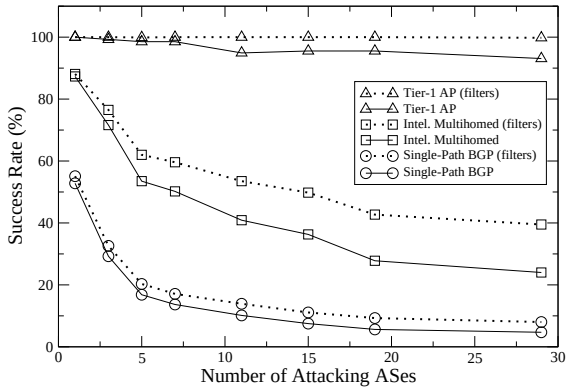


Figure 2: Success rate of sources reaching a hijacked destination when using different degrees of path diversity.

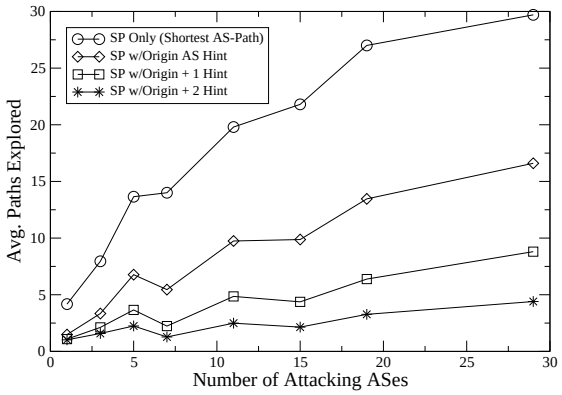


Figure 3: Number of routes explored before finding a valid forwarding path.

top line, far right). To defeat these adversaries, legacy ISPs can employ a tactic already common among large providers today: filtering routes from customers to accept only prefixes that the customers own and have registered. As a result, these filters block malicious advertisements by other customers. Unlike filtering to protect the legacy BGP system (which must be performed globally), these filters need only be applied locally by some of the valid destination's transit providers. The results of applying such filtering at the ISPs between the tier-1 AP and the destination are shown by the "filters" lines. The results show that *filters provide complete protection with a tier-1 AP, but provide only incremental benefit for intelligent multi-homing or single-path BGP*.

Result 3: The time to find a valid route is reasonable in the face of many adversaries, and simple connectivity hints from the destination further speed the process. Figure 3 shows the average number of paths a source must explore, averaged over all Tier-1 APs, without the benefits of destination filtering. The *Origin AS Hint* case assumes that the source knows the correct AS originating the prefix being probed, while *Origin + x Hint* indicates knowledge of all upstream providers up to *x* hops from the origin (see §3.4). Note that by not incorporating historical knowledge of working routes this analysis represents a scenario significantly more challenging than the likely common case.

Without external topology information, ACR explores paths based only on their AS-path length. ACR must test a few paths per attacker before finding a working path, which we feel is not unreasonable. However, guiding path selection with some prior knowledge of topology is more efficient, requiring probing only a few paths even for large numbers of attackers. The topology hints force an adversary to pad its AS path to include the correct topology, which makes the path longer and less attractive to the shortest AS-path heuristic. Using these heuristics, ACR helps reduce outages to short "hiccups" in connectivity experienced while it explores new paths.

6 DEPLOYABILITY

ACR emphasizes low barriers to adoption: ACR simplifies deployment because it does not require cryptographic hardware in routers and because the functionality needed for path deflections is already widely available. Robustness for applications already using SSL or IPSec could be deployed immediately, with no dependence on an AS-level PKI and address ownership registries.

ACR benefits from backward compatibility: Changing a critical part of the Internet infrastructure raises stability and reliability concerns. Because ACR runs alongside BGP, not as a replacement, operators can evaluate it on operational networks without the need for a parallel test infrastructure. Additionally, failures within ACR are isolated from BGP. As a result, unlike many secure replacements for BGP, legitimate use or misconfiguration of ACR is unlikely to result in worse reachability than is provided by legacy BGP, because the single-path legacy BGP route is still available for use.

ACR provides well-incentivized deployment: We envision deflection services being offered in two ways. First, core networks can offer deflections to their directly-connected transit customers. This could give an ISP a competitive advantage: customers will receive improved resilience against attacks and gain the ability to select paths that perform better.

The second deployment scenario is to offer a remote deflection service to ASes that are not direct transit customers. This service would enable customers of legacy ISPs to gain many of ACR's benefits. This remote deflection service is more technically challenging to offer, but as §5 showed, even deployment by a single large ISP can provide greatly improved attack resilience. An AP can offer remote deflection service more cheaply than normal transit service because (1) availability customers do not need a physical router port and (2) a tier-1 AP also re-

ceives more overall transit revenue because of increased traffic entering its network for deflections. As a result, stubs with both types of providers need not be "double-charged" for their connectivity.

7 Related Work

ACR is similar in spirit to seminal work performed by Perlman [13]. Secure routing has been pursued extensively in academia and industry; due to space constraints, we refer the interested reader to a recent survey of BGP security research [5]. ACR's path selection can benefit from secure routing protocols, but remains effective without them.

Popular current approaches for robust routing use overlay networks [2] or multi-home the edge [1]. While these techniques improve availability against many failures, we know of no studies that examine their resilience to deliberate routing attacks. Our evaluation suggests that they cannot withstand powerful adversaries that use BGP to globally disrupt routes to a destination.

Many clean-slate source-routing architectures either do not address security (e.g., NIRA [19]), or conflict with operational practices (e.g., feedback based routing [21]) by requiring the disclosure of routing policies often guarded today by non-disclosure agreements.

Recent work on router-level deflections [20] offers a complementary technique that provides finer-grained path diversity, but with less source control over how packets are deflected; ACR could leverage such techniques to help avoid adversaries within an AS.

8 Conclusion

ACR demonstrates that communication security can be achieved *without* securing the routing protocols. Because properties such as confidentiality and integrity can, and often already are, provided end-to-end by applications requiring strong security, this paper argues that availability is the only property that the routing system must provide. Availability, we believe, is better achieved by lightweight, incentive-compatible mechanisms to expose multiple paths to the network edge than by heavyweight secure routing techniques.

By recognizing that many applications today already require and use end-to-end security, ACR presents a novel and compelling point in the routing security design space. ACR demonstrates that robust routing and forwarding are in fact achievable given building blocks already common on the Internet today, and that the adoption of these mechanisms can occur in a well-incentivized and incremental way. Because ACR also provides strong protection from data-plane adversaries and failures, we believe its principles are a worthwhile addition to the routing security toolbox, regardless of whether a secure version of BGP is eventually deployed.

Acknowledgments

The Dept. of Homeland Security helped to fund this work with HSARPA grant 1756303 and a graduate fellowship for Dan Wendlandt. Special thanks to Adrian Perrig, Nick Feamster, our anonymous reviewers, and many others whose comments greatly improved this work.

References

[1] A. Akella, J. Pang, B. Maggs, S. Seshan, and A. Shaikh. A comparison of overlay routing and multihoming route control. In *Proc. ACM SIGCOMM*, Aug. 2004.

[2] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proc. 18th ACM Symposium on Operating Systems Principles (SOSP)*, pages 131–145, Oct. 2001.

[3] Arbor Networks. Arbor networks: Infrastructure security survey. http://www.arbornetworks.com/sp_security_report.php, 2006.

[4] I. Avramopoulos and J. Rexford. Stealth probing: Efficient data-plane security for IP routing. In *Proc. USENIX Annual Technical Conference*, May/June 2006.

[5] K. Butler, T. Farley, P. McDaniel, and J. Rexford. A survey of BGP security. Technical Report TD-5UGJ33, AT&T Labs, June 2004.

[6] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Comm. of the ACM*, 4(2), February 1981.

[7] B. Christian and T. Tauber. *BGP Security Requirements*. IETF, Apr. 2006. Internet Draft: draft-ietf-rpsec-bgpsecrec-06.txt.

[8] P. Francios and O. Bonaventure. An evaluation of IP-based fast reroute techniques. In *Proc. CoNEXT'05*, 2005.

[9] L. Gao. On inferring autonomous system relationships in the Internet. *IEEE/ACM Trans. Netw.*, 9(6):733–745, 2001.

[10] S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol (S-BGP). *IEEE JSAC*, 18(4):582–592, Apr. 2000.

[11] J. Lau, M. Townsley, and I. Goyret. Layer two tunneling protocol - version 3 (L2TPv3). RFC 3931, IETF, Mar. 2005.

[12] J. Ng. *Extensions to BGP to Support Secure Origin BGP (soBGP)*. IETF, Apr. 2004. Internet Draft: draft-ng-sobgp-extensions-02.txt.

[13] R. Perlman. Network layer protocols with bynzantine robustness. Technical Report TR-429, MIT LCS, Oct. 1988.

[14] A. Ramachandran and N. Feamster. Understanding the Network-Level Behavior of Spammers. In *Proc. ACM SIGCOMM*, 2006.

[15] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang. BGP routing stability of popular destinations. In *Proc. ACM SIGCOMM Internet Measurement Workshop*, Nov. 2002.

[16] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. Katz. Listen and Whisper: Security mechanisms for BGP. In *Proc. Symposium on Networked System Design and Implementation*, Mar. 2004.

[17] D. Walton, A. Retana, and E. Chen. *Advertisement of Multiple Paths in BGP*. IETF. Internet Draft: draft-walton-bgp-add-paths-05.txt, Expired August 2006.

[18] W. Xu and J. Rexford. MIRO: Multi-path interdomain routing. In *Proc. ACM SIGCOMM*, Sep. 2006.

[19] X. Yang. NIRA: A New Internet Routing Architecture. In *ACM SIGCOMM Workshop on Future Directions in Network Architecture*, Aug. 2003.

[20] X. Yang, D. Wetherall, and T. Anderson. Source selectable path diversity via routing deflections. In *Proc. ACM SIGCOMM*, 2006.

[21] D. Zhu, M. Gritter, and D. Cheriton. Feedback based routing. In *Proc. HotNets-I*, Oct. 2002.

A Technical Approach to Net Neutrality

Xiaowei Yang Gene Tsudik Xin Liu
Department of Computer Science
University of California, Irvine
[xwy|gts|xinl]@ics.uci.edu

ABSTRACT

A recent statement by AT&T CEO Ed Whitacre sparked considerable fear in the public that the Internet may not be open any more: the ISPs dictate which sites/applications flourish and which flounder. The statement triggered the heated debate on net neutrality and ignited the battle to enact net neutrality legislation. However, by the date of writing, all attempts to pass net neutrality laws have failed.

This paper states our proposition on net neutrality: ISPs should not be able to discriminate against packets based on contents, application types, or packet sources or destinations that are not their own customers; but they are eligible to offer differentiated services to their customers. We present a technical design that aims to achieve this definition of net neutrality. Our design prevents an ISP from deterministically harming an application, a competing service, or singling out an individual innovator for extortion.

1 INTRODUCTION

In November 2005, AT&T CEO (formerly SBC CEO) Ed Whitacre was quoted in **BusinessWeek** as follows [3]:

> "Now what they [Internet upstarts like Google, MSN, Vonage, and others] would like to do is use my pipes free, but I ain't going to let them do that because we have spent this capital and we have to have a return on it," says Whitacre. "So there's going to have to be some mechanism for these people who use these pipes to pay for the portion they're using. Why should they be allowed to use my pipes?"

This statement triggered strong reactions from consumers and Internet companies. A number of net neutrality draft bills [20] were introduced to Congress since March 2006, in attempt to enact net neutrality. Unfortunately, none has succeeded so far. Grassroot coalitions such as "Save The Internet" [18] and "It's Our Net" [13] were created. Hundreds of organizations and companies joined the coalitions, and more than a million signatures were collected to support net neutrality. Competitors such as Google, Yahoo, and Microsoft, grassroot groups from both the left and right (e.g. Moveon.org and Christian Coalition of America), stand on the same side to support net neutrality.

Proponents of net neutrality argue that the openness of the Internet, i.e., the ability to access any content, run any application, or attach any device to the Internet, leads to the very success of the Internet. This openness and freedom drives innovation, promotes free speech, and encourages democratic participation [4, 18]. If ISPs were able to discriminate packets based on content or ownership, innovative ideas would not necessarily be rewarded. Instead, well-funded ideas or ISPs' own services would be more likely to succeed.

Opponents of net neutrality (e.g. telcos) [11] argue that tiered service, or data prioritization, is a legitimate business model. The increasingly popular video and audio applications on the Internet put a high bandwidth demand on their networks. Tiered service can provide desired quality of service to different applications and recoup the capital investment used to upgrade their networks. Some opponents dislike the idea of regulation in principle, arguing that market forces are sufficient to regulate what broadband ISPs would do. If one ISP blocks contents or applications that consumers desire, consumers would switch to a different ISP.

Both sides have their points, which makes the debate over net neutrality a murky matter. On the one hand, data prioritization can improve quality of service and is a legitimate business model. A strict neutrality law that does not allow fee-based data prioritization, e.g. the Markey Amendment "Network Neutrality Act of 2006" [16], will prohibit ISPs from selling differentiated services, and prevent customers from purchasing improved quality of service based on their willingness to pay. On the other hand, ISPs may abuse data prioritization to discriminate packets to their favor. For instance, telcos may give a high priority service to their own VoIP service and intentionally slow down a competitor's service.

It is difficult to conclude whether any net neutrality law should or will be passed in the near future, partly because of the difficulty of line-drawing and the suspicion that there is sufficient market competition. A few pundits have advocated the wait-and-see approach to avoid potentially harmful or toothless regulations [10]. However,

the status quo without regulation has its own risk: the openness of the Internet may gradually erode and innovations may be stifled. It's true that there is some level of competition (i.e. with cable competing with DSL) in the broadband market, but practically speaking, users tend to stay with their existing service providers despite of mild service dissatisfaction for a number of reasons, e.g., costs of switching, bundling deals, or cancellation hassles [8]. A broadband ISP may take advantage of this inertia to gradually migrate to a closed Internet. As an example, a broadband ISP may intentionally degrade the VoIP service offered by Vonage, but give a high priority service to its own VoIP offerings. A user that experiences a low-quality VoIP service from Vonage but could use a substitute service offered by his provider might not bother to switch. Using this tactic, gradually, a broadband service provider may drive Vonage out of business and make its own VoIP service thrive. Then it can start to degrade another competitor's service and so on.

Alternatively, individual innovators that can afford to pay (say Google) might choose to pay every access provider to avoid appearing slow to users. However, it's unclear whether there is sufficient market force to regulate the price Google needs to pay, because once a user has chosen his access provider, that access provider becomes a monopoly to Google. There is no way for Google to bypass the access provider to reach the user.

In this paper, we propose a definition of net neutrality as follows: ISPs should not be able to discriminate against packets based on contents, application types, or packet sources or destinations that are not their own customers. We call this type of discrimination *non-neutral* discrimination. But ISPs are eligible to offer differentiated services to their customers. Our hypothesis is that the present market structure may not have sufficient competition to prevent an access ISP from degrading the service of a particular application or a site, but might be sufficient to keep them from intentionally ill-treating their own customers. For instance, if AT&T slows down a customer's VoIP traffic from Vonage, the customer may not care to switch to a different provider. But if AT&T slows down all traffic the customer sends or receives, or charges a higher price for the same quality of service than what the customer can get at a different provider, the customer may decide to switch. We rely on the existing market competition to regulate how ISPs treat their own customers or peers' traffic. If there turns out to be sufficient market competition, then ISPs would not overcharge their customers for the desired quality of service, and would strive to meet the service requirements of their customers. In this situation, a customer's traffic will not be intentionally harmed regardless of how ISPs prioritize their own services or other customers' high-quality services. If there is no sufficient competition, then consumers as a whole would suffer. Hopefully they as a whole are a stronger voice than individual innovators and this situation can make it clear what net neutrality regulation is needed.

This paper presents a technical design that aims to realize our definition of net neutrality, a solution that prevents non-neutral discrimination yet allows tiered services. A key design challenge is to prevent an ISP from discriminating against a source or a destination that is not its customer (or peer). Standard end-to-end encryption techniques (e.g., IPsec) can be used to prevent content-based or application-based discrimination, but the source or destination address of a packet may still reveal the identity of a non-customer. To address this challenge, we design an efficient and stateless neutralizer service that allows an ISP to "blur" packets to or from its customers. Thus, other ISPs cannot target an individual customer of the ISP and double charge the customer.

The rest of the paper is organized as follows. We describe our design assumptions and design details in § 2 and 3. § 4 provides a preliminary performance analysis. § 5 discusses related work, and we conclude in § 6.

2 ASSUMPTIONS

We assume that there are ISPs that support net neutrality, perhaps due to the competitive pressure in the backbone market or sharing the belief that an open Internet is the key to foster innovation. This assumption is not unrealistic. For instance, Cogent has made a public statement in support of net neutrality [6]. We assume such ISPs are willing to offer services to their customers to protect them from being double-charged by broadband access ISPs. We also assume that host software can be modified to support our design.

We refer to an ISP that intends to discriminate packets in a non-neutral manner as a discriminatory ISP. We assume that a discriminatory ISP, despite its eagerness to make money, will not launch active attacks at its customers or peers. Those attacks include modifying packet contents, man-in-the-middle attacks, and denial of service (DoS) attacks. The ISP may eavesdrop on all traffic, perform traffic analysis, delay or drop packets within its network, but it cannot eavesdrop on traffic outside its network. We believe this assumption is realistic because malicious behavior, once detected, may severely damage an ISP's reputation.

For simplicity, our current design does not consider traffic analysis attacks that infer application types or packet ownships using packet size and timing information. If in the practical deployment ISPs can use traffic analysis to successfully discriminate, we will consider incorporating mechanisms such as adaptive traffic masking [19] to defeat such attacks.

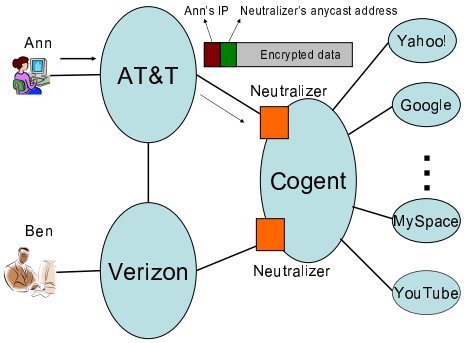


Figure 1: **The neutralizer boxes mix packets sent to and from Cogent's customers. Other ISPs such as AT&T and Verizon cannot differentiate its customers' packets.**

We assume each packet carries a standard IP header, and additional fields needed by our design are carried in a shim layer between IP and an upper layer. The protocol field in an IP header is set to a fixed and known value. The source and destination address fields refer to the fields in a standard IP header.

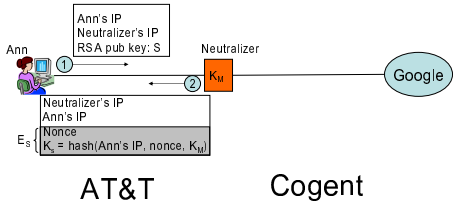
3 DESIGN

The goal of our design is to prevent an ISP from discriminating packets in a non-neutral manner. We use two techniques to accomplish this goal. One is the existing end-to-end encryption techniques. The other is a neutralizer service as described below.

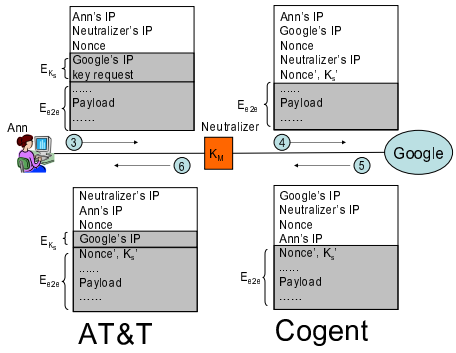
Figure 1 shows a high-level view of our design. A non-discriminatory ISP, Cogent in this example, places neutralizers at the boundary of its domain. These neutralizers can either be inline boxes or part of a border router's functionality. A neutralizer helps an ISP's customers to hide their addresses from other ISPs. In Figure 1, all packets sent to (or from) the customers of Cogent, e.g. Google, Yahoo!, MySpace, or YouTube , will have the neutralizer's IP address as their destination (or source) addresses. We use an anycast address to represent the neutralizer service of an ISP. All customers of an ISP use the same neutralizer address, regardless of where they are located. Further, end-to-end encryption is used to protect packet payload. With this design, a discriminatory ISP, e.g. AT&T in this example, can only discriminate against Cogent's packets as a whole, and cannot deterministically harm an individual customer of Cogent.

3.1 Bootstrapping

To bootstrap a connection, a source inside a discriminatory ISP needs to obtain a destination's IP address, the destination's neutralizers' addresses, and the destination's public key for end-to-end encryption. This bootstrapping information can be stored at a destination's DNS records, and a source may obtain this information via DNS queries.



(a) A user Ann in a discriminatory ISP sets up a symmetric key with a neutralizer.



(b) Ann sends and receives packets with encrypted addresses via the neutralizer.

Figure 2: **The shaded areas represent encrypted data, and non-shaded areas represent plain text. The letter E denotes the encryption function, and the subscript denotes the encryption key. K_M is the master key of a neutralizer. The first two fields in a packet diagram are the source and destination address fields. The rest of the fields are either in a shim header or payload. As can be seen, the address of Cogent's customer, Google's IP, and packet contents are not visible inside AT&T.**

A discriminatory ISP may eavesdrop on its customer's DNS queries and discriminate DNS queries based on the query destination. For instance, AT&T may delay queries for `www.google.com` if Google does not pay AT&T to use its pipes. To address this problem, a source needs to encrypt its DNS queries and send the queries to DNS resolvers that are not controlled by the discriminatory ISP. We assume a third party, such as a non-discriminatory ISP, a large overlay network, e.g. the PlanetLab [17], or Google itself, can provide DNS resolvers to clients inside discriminatory ISPs. Those clients will be configured with the IP addresses, the public keys, and the neutralizers' addresses (if there is any) of those DNS resolvers.

End-to-end encryption can use standard techniques such as IPsec. In this paper, we use end-to-end encryption as a black box, and do not discuss the details.

3.2 Efficient and Stateless Neutralizer

After a source obtains the bootstrapping information, it can send packets to a destination via the neutralizer. The source and the neutralizer must keep the destination address as a secret to avoid discrimination. This is sim-

ilar to anonymous routing [7]. However, a neutralizer in our context must be highly efficient and scalable, because it may receive all traffic sent to or from a large ISP at a peering point. Therefore we cannot use existing anonymous routing techniques [7, 14], that require per-flow state and expensive public key operations.

Our design uses a combination of light-weight public key encryption and symmetric key encryption to improve efficiency, and a technique that allows a neutralizer to compute a symmetric key from a packet header to avoid state. We cannot completely avoid public key encryption (or any technique that's equivalent) because protecting the secrecy of the destination address requires the establishment of a shared secret between a source and a neutralizer. It is a well-known result in theoretical cryptography that the establishment of a shared secret in the presence of an eavesdropper is impossible using only the techniques of symmetric-key cryptography [12].

Figure 2 illustrates how a source communicates with a destination via a neutralizer. The source first generates a short (e.g. 512-bit) one-time RSA public key S, and sends the key to the neutralizer. The neutralizer keeps a long term master key K_M. When it receives a public key from a source, it chooses a random nonce, and computes a keyed hash (K_s) from the nonce, its master key, and the source address: $K_s = hash(K_M, nonce, srcIP)$. K_s will be used as the symmetric key between the source and the neutralizer. The neutralizer encrypts the nonce and the symmetric key using the source's public key, and returns the encryption to the source. The source decrypts the packet, and retrieves the nonce and the symmetric key. It can then encrypt a destination address with the symmetric key and send the packet to the neutralizer. The source sends the nonce in clear text for the neutralizer to recover the shared key K_s. When the neutralizer receives the packet, it recomputes the symmetric key as $K_s = hash(K_M, nonce, srcIP)$ and decrypts the destination address.

This key setup process has the advantage that the neutralizer is stateless and performs the more efficient RSA encryption operation, while the source executes the slower RSA decryption operation. (An RSA encryption may involve as few as two multiplications, if the exponent in the public key is 3.) It also maintains the stateless and fault-tolerant feature of IP routing. As long as the neutralizers of a domain share the master key K_M, any neutralizer can decrypt the destination address and forward the packet.

A short RSA key represents a tradeoff between efficiency and security. A 512-bit RSA key is only as secure as a 56-bit symmetric key. To improve security, we let a source use a short RSA key only once, and expire the symmetric K_s key (encrypted with the RSA key) quickly. As shown in Figure 2, when a source sends the first packet to a destination using the the symmetric key K_s, it also sends a key request. When the neutralizer forwards the packet with a key request, it stamps a new nonce, and a new key K_s' into the packet. When a destination receives this packet, it uses strong end-to-end encryption, e.g. 1024-bit RSA encryption, to encrypt this new (nonce, key) pair together with its packet payload, and sends them to the source. A source can encrypt the destination address in its subsequent packets with this new key until it obtains a newer one. As long as a discriminatory ISP does not factor the short RSA key before K_s' is returned to the source (which takes two round trip times), the discriminatory ISP cannot decrypt the destination address, and cannot discriminate packets based on the destination address.

Another advantage of this design is that if a neutralizer cannot support RSA encryption at line speed, it can offload the encryption operation to any customer in its domain that is willing to help. The neutralizer inserts the nonce and the symmetric key K_s in the source's key request packet and forwards the packet to the customer to encrypt using the public key in the request packet. A customer (e.g. Google) would have incentive to help because the source may intend to communicate with it.

We have also considered a more obvious design choice that lets a source encrypt a destination address using a neutralizer's public key when it sends the first packet to a destination. This alternative has the advantages of saving one round trip time for key setup and preventing the man-in-the-middle attack, as the public key of a neutralizer can be certified. However, it places a higher burden on a neutralizer: the neutralizer must perform a public key decryption operation that cannot be offloaded to any customer.

We chose the first key setup approach because a higher processing overhead makes a neutralizer more vulnerable to DoS attacks and temporary traffic overloading, while an extra round trip time for key set up can be amortized over multiple packets. A source can use the same symmetric key to send any packet destined to any customer in the neutralizer's domain until the neutralizer's master key expires. We also assume that a discriminatory ISP will not risk its reputation to launch man-in-the-middle attacks. Thus, the second advantage of the alternative approach is not essential.

A return packet from a customer in a neutralizer's domain must have its source address anonymized, yet the recipient must be able to retrieve the necessary key to decrypt the payload of the packet. In our design, when a destination returns a packet to the source, the return packet will include the destination's address in the source IP address field, the neutralizer's address in the destination IP field, the initiator's address and the nonce included in the forward packet in a shim header, as shown

in Figure 2. When a neutralizer receives a return packet from a customer (we assume the neutralizer can tell this from the source address field), it encrypts the source address (Google's IP in Figure 2) using the symmetric key indicated by the nonce, replaces the source address with its own anycast address, and sets the destination address to be the initiator's address. When the initiator receives this packet, it can use the nonce and the neutralizer's address to locate the key K_s it shares with the neutralizer, and decrypt the original source address (Google's IP). It can then use this address to identify the communication session, and decrypt the packet payload.

3.3 Reverse-direction Communication

Communication initiated by a customer (e.g. Google in Figure 2) inside the neutralizer's domain to an outside destination can happen in a similar fashion but with less overhead. In the initial key set up phase, the customer may simply request a nonce and a symmetric key from a neutralizer without encryption. After the customer obtains a shared key with a neutralizer, the rest of the process is similar to what we have described above. The customer encrypts the shared key with its intended destination's public key and sends the encrypted key. When the destination, e.g. Ann's computer in Figure 2, receives a packet, if it cannot locate a symmetric key corresponding to the nonce and the neutralizer's address, it will attempt to use its public key to decrypt the packet. If successful, the destination obtains the shared key and can use it to encrypt the initiator's address and send packets via the neutralizer.

3.4 Quality of Service

In our design, a discriminatory ISP can still offer differentiated services [1] to its customers, as a neutralizer will not modify the Differentiated Services Code Point (DSCP) in a standard IP header. The discriminatory ISP may provide differentiated services according to the DSCPs in packet headers.

However, a discriminatory ISP can no longer keep per flow state (a flow refers to a source and a destination pair) to provide guaranteed services [2] to anonymized traffic. There are at least two remedies to this problem. The first is for a neutralizer to assign a dynamic address to a customer that initiates a QoS session, e.g. an RSVP session. This dynamic address allows the discriminatory ISP to identify a flow, but does not allow it to map the flow to a specific customer. Another possibility is that the customer may request not to be anonymized if it has purchased guaranteed service from the discriminating ISP. The neutralizer service is optional, and a customer does not have to use it.

3.5 Multi-homed Sites

A site may connect to multiple ISPs that offer the neutralizer service. In this situation, the site may publish multiple neutralizers' addresses in its DNS records, each address corresponding to one ISP. The ISP-level path of the site's incoming and outgoing traffic is then controlled by how other sources pick the neutralizers, and is no longer controlled by the site's BGP routers. The path chosen by other sources may interfere with a site's traffic engineering effort. For instance, if one provider is congested, the site may want all traffic comes from a different provider, but other sources may choose the congested provider. A similar situation occurs in IPv6 [9], in which a multi-homed site obtains multiple addresses, one from each provider. The path of the incoming traffic to a site is determined by which address a source chooses to contact the site. We can borrow any technique that can balance traffic load in that context to balance traffic between different neutralizers. In general, two hosts may always use trial-and-error to find a path that's working for them.

3.6 What Can Still Go Wrong?

Denial of Service Attacks: A neutralizer box may be subject to DoS attacks. Although our design places the more efficient RSA encryption operation at a neutralizer, a public key operation is still expensive. If attackers flood key setup packets at line speed, a neutralizer may be overloaded. It is outside the scope of this paper to come up with a complete DoS defense mechanism. But a neutralizer can invoke DoS defense mechanisms such as pushback [15] to get rid of attack traffic.

One complication is that if attackers are inside a neutralizer's domain, the neutralizer's anonymization function may hide attack sources. This problem is similar to source address spoofing. Pushback can be used to defend this type of attack, as it is designed to function well with source address spoofing and does not rely on source addresses to filter attack traffic.

Packet discrimination: Our design does not completely "blur" all packets. A discriminatory ISP can still discriminate packets in at least three ways: 1) discriminate based on its customers' or neutralizers' addresses; 2) discriminate against encrypted traffic; 3) discriminate against key setup packets. (The third discrimination is possible because an ISP may infer a key setup packet from the nonce field, or from the packet length, or from inter-packet timing.) We are not concerned with these types of discriminations because none of them allows an ISP to deterministically harm an application, a competitor's service, or a non-customer/peer. If an ISP can only deterministically discriminate against its own customers (or its direct peers), then we rely on market forces to discipline what they would do (§ 1).

Good-intentioned discrimination: If packets are not encrypted or neutralized, an ISP may inspect packet contents and prevent unwanted traffic (e.g. viruses) from reaching an end user. Unfortunately, our design prevents such good-intentioned discrimination. Nonetheless, we believe it is a worthy tradeoff, because we cannot afford to lose the openness of the Internet.

4 PRELIMINARY EVALUATION

This section presents a preliminary evaluation on the performance of the neutralizer. We implemented the packet processing logic of a neutralizer using a modified Click Router [5] on Linux 2.6.16.13. The neutralizer in our testbed has an AMD Opteron 2.6GHz dual core CPU and an Intel pro/1000 GT quad-port server adapter.

A neutralizer does an RSA encryption when processing a key setup packet. In our experiments, the neutralizer can output response packets at 24.4kpps. If we assume a neutralizer's master key lasts for an hour, a source outside a neutralizer's domain at most needs to send a key request once an hour. Thus, a commodity PC can simultaneously serve 88 million sources for key setup.

A neutralizer does a hash computation and a symmetric key encryption or decryption when it receives a normal data packet. Our implementation uses 128-bit AES for both hashing and encryption/decryption. In our experiments, a client machine sends neutralized UDP packets with 64 bytes payload to the neutralizer. The total packet size is 112 bytes after adding headers, nonce, encrypted destination IP address, and alignment padding. The neutralizer is able to output packets with decrypted destination IP addresses at 422kpps. This throughput is limited by the neutralizer's hardware architecture, as the neutralizer can only forward vanilla IP packets of the same size at 600kpps. Our openssl speed tests show that the CPU of the neutralizer can perform the cryptographic operations at 2.35 million per second. We expect that special hardware that is optimized for packet forwarding can achieve a much higher packet throughput.

5 RELATED WORK

The most related work is anonymous routing [7, 14]. Anonymous routing aims to anonymize both the source and destination addresses of a packet, while our design only aims to anonymize the non-customer address from a discriminatory ISP. As a result, our design is considerably more efficient and scalable in terms of resource consumption. In our design, routers don't keep per-flow state, and perform much fewer public key encryption/decryption operations.

6 CONCLUSION

The debate over net neutrality has caught much public attention recently. Despite much effort, no essential net neutrality legislation is passed. This paper presents a technical approach to net neutrality. We describe a design that prevents ISPs from discriminating packets based on contents, application types, or non-customer/peer addresses. Our design prevents an ISP from deterministically discriminating against a competitor's service, a novel application, or singling out an individual innovator for extortion. At the core of our design is a neutralizer service that anonymizes traffic sent to or from a discriminatory ISP. The neutralizer is stateless and uses highly efficient cryptographic operations. We believe it can scale to support the traffic load of a large ISP. It's our future work to implement the neutralizer service and evaluate its performance.

REFERENCES

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. Internet RFC 2475, 1998.

[2] R. Braden, D. Clark, and S. Shenker. *Integrated Services in the Internet Architecture: an Overview*. IETF, June 1994. RFC 1633.

[3] BusinessWeek. Online Extra: At SBC, It's All About "Scale and Scope". `http://www.businessweek.com/magazine/content/05_45/b3958092.htm`.

[4] Center for Democracy & Technology. Preserving the Essential Internet. `http://www.cdt.org/speech/20060620neutrality.pdf`, June 2006.

[5] Click router. `http://www.read.cs.ucla.edu/click/`.

[6] Cogent Supports Net Neutrality. `http://www.cogentco.com/htdocs/neutrality.php`.

[7] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of USENIX Security Symposium*, 2004.

[8] B. Dipert. Net neutrality: Permissible restrictions. `http://www.edn.com/blog/400000040/post/1720005172.html`, Nov. 2006.

[9] R. Draves. *Default Address Selection for Internet Protocol version 6 (IPv6)*, 2003. RFC 3484.

[10] E. W. Felten. Nuts and Bolts of Network Neutrality. `http://itpolicy.princeton.edu/pub/neutrality.pdf`.

[11] Hands off the Internet. `http://www.handsoff.org/`.

[12] R. Impagliazzo and S. Rudich. Limits on the Provable Consequences of One-Way Permutations. In *Proc. of 21st Annual ACM Symposium on the Theory of Computing*, pages 44–61, 1989.

[13] It'sOurNet. `http://www.itsournet.org/`.

[14] S. Katti, D. Katabi, and K. Puchala. Slicing the Onion: Anonymous Routing without PKI. In *ACM HotNets*, College Park, MD, November 2005.

[15] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling High Bandwidth Aggregates in the Network. *Computer Communications Review*, 32(3), July 2002.

[16] E. Markey. Network Neutrality Act of 2006. `http://markey.house.gov/docs/telecomm/Markey%20Net%20Neutrality%20Act%20of%202006.pdf`, May 2006.

[17] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *Proc. of HotNets-I*, Oct. 2002.

[18] Save the Internet. `http://www.savetheinternet.com`.

[19] B. Timmerman. A security model for dynamic adaptive traffic masking. In *Procs of the 1997 workshop on New security paradigms*, 1997.

[20] Network Neutrality. `http://en.wikipedia.org/wiki/Net_neutrality`.

An Axiomatic Basis for Communication *

Martin Karsten
University of Waterloo
mkarsten@uwaterloo.ca

S. Keshav
University of Waterloo
keshav@uwaterloo.ca

Sanjiva Prasad
IIT Delhi
sanjiva@cse.iitd.ac.in

ABSTRACT

The de-facto service architecture of today's communication networks lacks a well-defined and coherent theoretical foundation. With layering as the only means for functional abstraction, the diversity of current technologies cannot be expressed consistently and analyzed properly. In this paper, we present an axiomatic formulation of fundamental mechanisms in communication networks. In particular, we reconcile the existing but somewhat fuzzy concepts of *naming* and *addressing* and present a consistent set of primitives that are sufficient to compose communication services. The long-term goal of this exercise is to better document, verify, evaluate, and eventually implement network services.

1 INTRODUCTION

Traditionally, the Internet is modelled as a graph, where each node implements a set of protocol layers and each edge corresponds to a physical communication link. Unfortunately, when compared to the actual Internet, this model falls far short. In the traditional model, nodes are addressed by one or more static IP addresses. End systems implement a simple five-layer stack, with applications using a transport layer to access IP, which is layered on the data link and physical layers. Packet forwarding decisions are made purely on the basis of IP 'routing' tables. Moreover, a protocol layer at any node only inspects packet headers associated with that layer, obeying strict layering rules in dealing with other layers. In reality:

- DHCP, anycast, multicast, NAT, mobile IP and IP tunnelling break the static association between a node and its IP address.
- Nodes implement more layers, including IP or VLAN tunnels, overlays, and shims, e.g. MPLS.
- Forwarding decisions are made not only by IP routers, but also by VLANs, MPLS routers, NAT boxes, firewalls, and mesh routing nodes.
- Middleboxes and cross-layered nodes such as NATs, firewalls, and load balancers violate layering.

In face of these significant extensions to the classical model, understanding the topology of the Internet in terms of its connectivity has become a daunting task. It has become difficult to even define elementary concepts such as a neighbour and peer relationships, let alone the more complex processes of forwarding and routing. Further, there is not even a common and well-defined language for fundamental networking concepts, with terms such as 'name', 'address', or 'port' being the subject of seemingly endless debate.

Yet, surprisingly, the system still works! Most users, most of the time, are able to use the Internet. What lies behind the unreasonable effectiveness of the Internet? We postulate that there are a set of underlying principles that are obeyed by extensions to the traditional model, no matter how ad hoc, which preserve connectivity. However, these principles have rarely been systematically studied (with [3, 4] being notable exceptions).

Our research goal, over the long term, is to axiomatically specify basic Internet concepts that allow us to construct (a) a theoretically sound framework to express architectural invariants–such as the deliverability of messages–even in the presence of network dynamism, middleboxes, and a variety of compositions of different protocols, and (b) an expressive pseudo-language in which to rapidly implement a variety of packet forwarding schemes. Therefore, the concepts, and the pseudo-language derived from them, serve not only to clarify the essential architecture of the Internet, but also provide a bridge between formal proofs on node reachability using a particular forwarding scheme, and a practical implementation of that scheme. Our goals are inspired by Hoare's axiomatic basis for programming [6].

In this paper, we take a first step in this direction by presenting an axiomatic framework of communication concepts and pseudo-language primitives derived from these concepts. We sketch how the framework can be formalized, but we do not discuss the implementation of the pseudo-language primitives. However, it will become clear that the primitives can be implemented in any reasonable packet forwarding engine.

To keep the problem tractable, we propose to split overall communication functionality into two broad areas: one area is concerned with *connectivity*, i.e. naming, addressing, forwarding, and routing. The second is the set of mechanisms to provide additional functionality related to communication quality and performance. This includes medium access control, reliability, flow control, congestion control, security, among others, and is not yet

*Supported by the National Science and Engineering Research Council of Canada, the Canada Research Chair Program, and Intel Co.

explicitly considered in our work.

Our work draws from, and is related to, a handful of other attempts to bring clarity to Internet architecture. Clark's seminal paper [3] succinctly laid out the design principles of the classical Internet, but does not provide a basis for formal reasoning about its properties. Recently, Griffin has used formal semantics to model routing [4] and Loo et al. have used a declarative approach to describe routing protocols [8]. Our work is directly related to past work in the area of naming and addressing indirection. This has been considered both in existing technology standards, such as IP Multicast, IPv6, or Mobile IP, as well as research proposals [2, 5, 9, 12]. Similar to our work, these proposals blur the traditional distinction between naming and addressing, and also consider innovative packet forwarding mechanisms. However, to our knowledge, these past proposals are essentially ad hoc, without a consistent set of underlying formal principles. In contrast, we suggest an axiomatic formulation of communication principles and thereby present a first attempt at building a complete formal basis for reasoning about communication systems.

2 Conceptual Framework

2.1 Naming and Binding

Naming and binding in computer systems is relatively well understood. Before introducing a new set of definitions, we first review and summarize some fundamental concepts from the seminal paper by Saltzer [11] with a few minor modifications, as noted.

- An *object* is a software or hardware structure in a computer system.
- A *name* is a regular expression that is used to refer to a set of objects. This is an extension from the original definition [11], which only refers to one string and one object. We use regular expressions to allow for wildcard matching and refer to a set of objects because of broadcast, anycast, and multicast communication styles.
- The original term *binding* [11] is used both as a noun, describing the existence of a mapping from a name to a set of objects, as well as a verb, referring to *choosing* the appropriate objects for a name. To avoid confusion, we use the term *mapping* when referring to the noun. In a traditional naming system, the single object can be accessed through a "lower-level" name [11], which is often called "address".
- A *context* is a set of mappings. A name is always interpreted relative to some context. To know the "lower-level" name associated with a name, one needs to also know which set of mappings is being referred to, because multiple contexts may provide different mappings for the same name.
- The *resolution* mechanism locates the appropriate mapping for a name in a particular context. This allows for access to the object through the corresponding "lower-level" name. The original definition is "locating the object" [11], which is identical to locating the mapping and accessing the object.

With these definitions, it is possible to develop the description of a basic naming system. However, this set of definitions stops at the concept of a "lower-level" name and simply assumes that it can be used to access a certain object. In a distributed system, however, the communication necessary to access a certain object is non-trivial and greatly influences the overall system behaviour.

2.2 Communication Concepts

Similar to Saltzer's usage of a "lower-level" name as primitive, we assume that certain objects can *directly communicate* with each other, without giving a formal definition of "direct communication". Direct communication is facilitated either by shared memory or takes place between low-level network entities that can directly exchange information via a physical medium, such as cable, radio, or fiber, for example in a local area network such as Ethernet. Based on this premise, we introduce the following definitions:

- We define a *network processing object (NPO)* as an object that can directly communicate with other NPOs. An NPO is an abstraction of a traditional protocol layer instance. Like any object, an NPO can be referred to by one or multiple names.
- An NPO may have a set of mappings associated with it, which is then called its *context state*. The set of mappings comprising the context state may contain wildcards. An example of a context state is a routing (or forwarding) table.
- NPOs that can directly communicate with one another are termed *neighbours*. An NPO can directly communicate with each of its neighbours using the neighbour's name. Examples of neighbouring NPOs are the TCP and IP NPOs on the same machine, or two MAC-layer NPOs on the same shared medium.
- The unit of communication is a *message*, which contains control information in the *header* and arbitrary data in the *payload*. The header might be explicit, as in a traditional packet header, or implicit, for example the time slot within a TDM frame during which a message is transmitted.

- A name that is encoded in the header of a message is termed an *address*. The message header contains, among other control information, a *stack* of addresses. The top-most address on the stack is the *destination address*.
- We define *forwarding* as an extension of direct communication, where NPOs repeatedly pass on a message to a set of neighbours, such that the message eventually arrives at a set of remote NPOs. In this sense, forwarding is the transitive relation of direct communication, necessary because not all NPOs are each others' neighbours.
- The original definition of *resolution* [11] needs to be generalized in that the result is not only a "lower-level" name, but includes further information to forward a message towards the set of NPOs referred to by a name.

Note that we have a particularly simple definition of an address: it is just a name that happens to be in a packet header and is therefore used to make a forwarding decision. This operational approach to defining an address bypasses myriad conceptual difficulties of other approaches. One immediate conclusion from this approach is that a name only needs to have local (per-NPO) syntax, while each address format must be standardized between NPOs. Note also that we explicitly describe each message as having a stack of addresses. When reading from and writing to the stack of addresses, multiple addresses may be transformed into one local name and vice versa. This allows us to model non-layered (or layer-violating) NPOs, such as middleboxes. For example, NAT operates on five address fields that internally are considered a single name.

2.3 Communication Operations

We define the local context state as the set of mappings from a name to a set of tuples of NPO and name as {<name → {<NPO, name>}>}.
The generic forwarding algorithm can then be described with the following pseudo-code using the primitives *send*, *receive*, *copy*, *push*, *pop*, *lookup*:

```
message msg = receive();
name n = pop(msg);
{<NPO, name>} S = lookup(n);
for each <NPO, name> s_i in S
  outmsg = copy(msg);
  push(outmsg, s_i.name);
  send(s_i.NPO, outmsg);
endfor
```

The `push` and `pop` primitives are specific to an NPO class and transform between a prefix of the address stack and a local name. Note that the above processing steps cover both ingress and egress processing for each NPO. Also, an NPO typically provides a *default mapping* which is used for all those names that do not have an explicit mapping in the context state. For example, in case of IP routing, this is called *default routing entry*.

The concepts and primitives introduced so far allow for the description of static communication scenarios, where forwarding tables and topologies do not change over time, and where local context state is sufficient to determine the neighbouring NPOs to whom a message should be forwarded. As an example, consider an IP network with pre-configured routing tables, running over Ethernet with all ARP lookups also pre-configured in the ARP cache.

2.4 Structure Concepts

The following definitions extend the basic communication concepts and allow to describe network structure at the familiar level of nodes and links.

- The NPO that inserts an address into a message header (by a `push` operation) along with those NPOs that *potentially* resolve the same address (using `lookup`) or remove it (`pop`) are termed *peers*.
- The communication association between a peer that writes a destination address into a message and a set of corresponding peers that receive the message and logically remove the destination address from the message (so that it it is no longer used for making a forwarding decision) is termed *link*. The sequence of peers forming a link is termed *path*.

Note that links are between peers. In contrast, neighbouring NPOs communicate via direct communication. Links are similar to ISO protocol interfaces, whereas direct communication refers to service interfaces. For example, an IP sender, IP router, and IP destination are peers, but not neighbours. A pair of connected Ethernet NICs can be considered as both neighbours and peers.

Using the concepts introduced so far, it is possible to describe data path mechanisms of a communication network. For example, we can talk of a link provided by two TCP NPOs that is established by a three way handshake. Similarly, a transient HTTP link exists between a browser client and a web server for the duration of the TCP link between them. An HTTP load balancer that examines the HTTP header would be a peer of the browser, and it would also be a peer to the web server. In this sense, the load balancer is a forwarding engine, on par with an IP router.

If suitable context state exists in all NPOs along a path, the message state necessary for forwarding a message to a set of remote NPOs can be reduced to a single name. Then, forwarding can be regarded as *binding* the name to

the set of destination NPOs. Based on this understanding, the following definitions provide concepts for structural properties.

- A set of peers that forward messages with the same destination address to the same set of NPOs provide *consistent* binding for this name.
- A *scope* for a set of names is a set of peers that provide consistent binding for each of these names. For each name in a scope, there is a unique sink tree leading to the NPO holding the corresponding mapping in its local context state. The full tree can be regarded as a distributed mapping.
- There can be special names in a scope, for example *broadcast* referring to all peers in a scope.
- Mechanisms and algorithms used to achieve consistency in a scope are collectively termed *routing*.

We assume that each individual NPO forms a natural scope for all names covered by its context state. Distributed control state such as distributed IP routing state can be described as a set of collaborating NPOs that form a scope for a set of names.

2.5 Distributed Resolution

The lookup primitive in Section 2.3 is defined on an individual NPO's context state and as such, inherently bound to a single object. A *context* as defined by Saltzer [11] is an abstract concept and not bound to any particular resource. Likewise, *resolution* is also not defined as local or distributed. While we assume that Saltzer implicitly refers to a local system only, the concepts work well in a distributed system.

Distributed resolution consists of forwarding a resolution request within the scope of the name to an NPO that has a mapping for this name and sending back the appropriate response. In a sense, the forwarding part is very similar to the definition of *closure* in Saltzer's work [11], which is defined as "the mechanism that connects an object wishing to resolve a name to a particular context".

For most scenarios, the definition of consistent binding nicely carries over to distributed resolution and resolution can be considered as binding a name to the appropriate object in a particular (distributed) context. The one exception is given by an anycast name that maps to multiple objects, but only a subset (typically one) of them is needed to successfully complete the task. In this case, consistency only applies to the set of eligible objects. In fact, caching in naming systems, being a special variant of replication, can be considered as anycast where the requested name can be bound to any of the available replicated objects. We note that implementations of anycast either rely on binding to some form of rendezvous point or employ a simplistic algorithm that does not change the original binding to a specific object, but shortcuts the resolution whenever possible, if a cached replica is found.

2.6 Control Operations

In this section, we introduce basic primitives that facilitate the interaction between control and communication operations. For simplicity and clarity, we do not model the algorithmic part of a control regime, for example distributed routing. Also, we ignore any access control that is necessary to validate control operations in reality.

Control operations are either triggered by special control messages (e.g. virtual circuit setup or routing), or implicitly depending on the message header (e.g. NAT setup). The corresponding details are beyond the scope of this paper. We only sketch the basic primitives that we envision to model control operations:

- `update(name, {<NPO, name>})`
  This primitive is used to add, remove, or update a mapping in context state.
- `create(name, op, {<NPO, name>})`
  This primitive creates and returns a control message containing the given arguments. The control message can be transmitted using the available forwarding primitives. The `LOOKUP` and `UPDATE` opcodes trigger the corresponding lookup or update operations at the destination NPO. The `INFO` opcode is used to communicate context state information to neighbours, for example routing information or name resolution replies.
- `control(name, op, {<NPO, name>})`
  This primitive represents the main entry point for control operations. For explicit control messages, the message content (cf. `create`) is passed and interpreted. For implicit control triggers, only the `name` is being used to determine the appropriate control operations. The algorithmic part of any control activity is also abstractly represented by this primitive. A detailed analysis of control operations is the subject of an ongoing study and will be presented in later work.

We use NAT as an example scenario between three nodes, as sketched in Figure 1, to illustrate the operation of and interaction between communication and control primitives. We ignore the ARP and outgoing Ethernet details to keep the example small and only show the processing inside the NAT node. Further, the execution of `pop` and `send` primitives is omitted from the example, since they are obvious. We use `UDP` as the local name of the UDP NPO instance at each node (aka "protocol number") and `IP` as the corresponding name for the IP

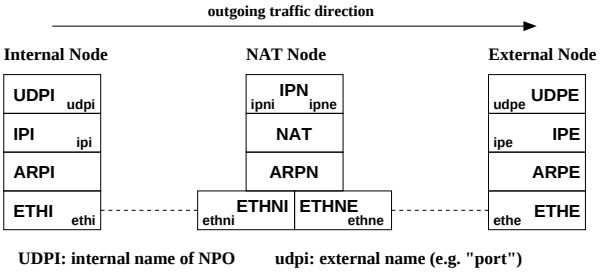


Figure 1: NAT Example Setup

instance. The address stack top to bottom is shown from left to right and we assume the convention that source addresses are pushed before destination addresses. We show the creation of a new UDP mapping:

```
ETHNI: lookup([ethni,ethi,IP])
NAT: control([ipe,ipi,UDP,udpe,udpi],
  NULL, <NULL, NULL>)
NAT: update([ipe,ipi,UDP,udpe,udpi],
  <IPN, [ipe,ipne,UDP,udpe,udpX]>)
NAT: update([ipne,ipe,UDP,udpX,udpe],
  <IPN, [ipi,ipe,UDP,udpi,udpe]>)
NAT: lookup([ipe,ipi,UDP,udpe,udpi])
NAT: push([ipe,ipne,UDP,udpe,udpX])
IPN: lookup([ipe,ipne,UDP])
IPN: push([ipe,ipe,ipne,UDP])
ARPN: lookup(ipe)
ARPN: push([ethe,ethne,IP])
ETHNE: ...
```

The Ethernet NPO at the incoming NIC performs a lookup for the destination address and the protocol field to determine whether the Ethernet frame should be received by this station and if yes, where to forward it to. The NAT NPO detects that the packet is sent from a non-local to an external address, but only finds the default mapping for this case, which invokes control processing. First, a free local UDP port `udpX` is determined, which is then used to create incoming and outgoing NAT context state. Afterwards, communication operations continue: The packet header is modified and the packet leaves via IP, ARP, and Ethernet NPOs.

The routing and forwarding of control requests and replies can be accomplished using available communication and control primitives. For example, the routing of reply messages, e.g. for resolution requests, is comparable to a NAT NPO that creates forwarding state for an outgoing messages and automatically routes the corresponding incoming messages.

3 FORMALIZATION

Formalization of the concepts introduced in the previous section provides a basis for rigorous analysis, validation and even formal verification of protocol design. We wish to operate at suitable levels of abstraction, and support modular analysis and refinement of specifications and formalizations. In this section, we sketch how a formal basis for our framework can be defined, and how formal analyses can be carried out.

3.1 Correctness Specifications

Consider the canonical requirement of *deliverability* of messages. This is easily specified as an *inductive* property of *a message msg at a given NPO np being deliverable to its destination NPOs*: (1) A message already at its destination NPO is deliverable. (2) A message *msg* at a particular NPO *np* is deliverable if *from each neighbour* np_i in mapping *m*, obtained from looking up the destination name, message msg'_i, constructed as specified by *m*, is deliverable. This is formalizable in logic as a predicate *msg@np deliverable* using inductive inference rules. Inference rules with no assumptions are called *axioms*. Predicates defined in such a manner are amenable to inductive proof techniques, with automated theorem prover support.

3.2 Operational semantics

We formalize the operational semantics of constructing and deconstructing messages, and manipulating context state as an *abstract machine* in the style of [7] running at each NPO. Configurations of the abstract machine are triples $\langle S|cs|p\rangle$ consisting of (i) a stack of values manipulated by the NPO (names, messages, mappings,..), (ii) context state, and (iii) sequence of primitive operations to execute. This formalization has a well-understood theory [10] and mechanizable reasoning apparatus, which permits the use of algebraic analysis techniques.

The low-level transition relation $\longrightarrow$ describes the change in configuration. Arguments to the operations are implicitly specified; they are at the top of the stack in the "pre" configuration, and in the "post" configuration, the results of the operation are at the top of the stack. In each rule, the first primitive operation of the third component is executed; the remaining operations are subsequently performed from the "post" state onwards.

$$\langle (n_1 n_2 ..., d)...|cs|\texttt{pop}; p'\rangle \longrightarrow \langle \{n_1, (n_2 ..., d)\}...|cs|p'\rangle$$
$$\langle \{(n_2 ..., d), x\}...|cs|\texttt{push}; p'\rangle \longrightarrow \langle (x n_2 ..., d)...|cs|p'\rangle$$
$$\langle n...|cs|\texttt{lookup}; p'\rangle \longrightarrow \langle m...|cs|p'\rangle$$
provided *cs* associates *m* to name *n*
$$\langle \{n, m\}...|cs|\texttt{update}; p'\rangle \longrightarrow \langle ...|cs[n \mapsto m]|p'\rangle$$
$$\langle \{n, m\}...|cs|\texttt{create}(o); p'\rangle \longrightarrow \langle ctl(n, o, m)...|cs|p'\rangle$$

Execution of `pop` expects a message of the form $(n_1 n_2 ..., d)$ on the stack, from the header of which the leading name is removed, returning a pair consisting of this name and the remainder of the message. Recall that the format of names is specific to the NPO, and so n_1 can

be a suitable prefix of the address stack, not merely the topmost address. Executing `push` expects a message and a name x, which is prepended to the message header to yield the resulting message. The `lookup` operation expects a name n on the stack, and the mapping associated with it in context state cs is returned. The `update` primitive expects a name n and a mapping m on the stack, which it uses to change the context state. The notation $cs[n \mapsto m]$ describes the context state that is identical to cs except that now name n is associated with mapping m. The `create` primitive takes an explicit opcode argument o, and expects a name n and mapping m on the stack. The result is a constructed message $ctl(n,o,m)$ which is placed on the stack for further processing.

The opcodes with non-local effect are those for communication. These may be described by rules that describe transfer of a message at one NPO to another:

$$np_1[\langle msg...|cs_1|\texttt{send}(np_2);p_1'\rangle],\ np_2[\langle ...|cs_2|\texttt{receive};p_2'\rangle] \longrightarrow np_1[\langle ...|cs_1|p_1'\rangle], np_2[\langle msg...|cs_2|p_2'\rangle]$$

provided np_1, np_2 can communicate directly. The notation $np[\ldots]$ indicates the state at NPO np, and in the rule the two communicating NPOs are juxtaposed. In this rule, we have presented a synchronous transfer of the message between the NPOs. However, for other semantics, we can interpose a suitable abstract medium with appropriate semantics (synchronous/asynchronous, queue/bag, lossy/ideal), which can be modelled either abstractly or explicitly.

3.3 Proof techniques

We separate notions of *partial correctness* or "safety" from those of termination or *progress*. The former properties describe that nothing wrong happens during protocol execution, e.g., that no message is incorrectly forwarded to an unintended NPO, or that no name is interpreted in an incorrect context. These are fairly challenging to establish in the presence of dynamic changes to the routing and forwarding state of the network [1], and require *coinductive techniques* to show that essential structural properties of the context state are maintained, i.e., are *invariant*. A typical invariant is that the context states for forwarding eventually form an acyclic directed graph. Other important properties that need to be shown are that updates due to messages maintain the necessary consistency of how names are interpreted in the context state. Another typical requirement is the existence of default forwarding actions which ensure deliverability.

We posit that the notion of scope will be useful in establishing invariants for proving the correctness of various protocols. For example, in IP mobility support, the scope corresponding to a home "IP address" may be considered as comprising those NPOs which will resolve this name to the correct current location of the device. While this set of NPOs can dynamically change due to mobility, the abstractly characterized scopes satisfy invariant properties such as: (i) they include the router at the 'home address"; (ii) that IP messages reaching a member of this scope remain confined to it; and (iii) that the forwarding tables will always take a message to a member of the abstractly characterized scope.

4 Conclusions

We believe that a carefully chosen conceptual framework of communication primitives not only provides a clear understanding of the current, complex Internet, but also serves as the basis for a formal model of its semantics. We have presented such a framework and outlined its operational semantics. In future work, we propose to further extend this formal analysis, and also implement a 'universal forwarding engine' based on our primitives.

References

[1] R. M. Amadio and S. Prasad. Modelling IP Mobility. *Journal of Formal Methods in System Design*, 17(1):61–99, 2000.

[2] H. Balakrishnan et al. A Layered Naming Architecture for the Internet. In *Proceedings of SIGCOMM 2004*, pages 343–352.

[3] D. Clark. The Design Philosophy of the Darpa Internet Protocols. In *Proceedings of SIGCOMM 1988*, pages 106–114.

[4] T. G. Griffin and J. L. Sobrinho. Metarouting. In *Proceedings of SIGCOMM 2005*, pages 1–12.

[5] M. Gritter and D. Cheriton. An Architecture for Content Routing Support in the Internet. In *Proceedings of USITS 2001*, pages 37–48.

[6] C. A. R. Hoare An Axiomatic Basis for Computer Programming. *Communications of the ACM*, 12(10):576–580, 1969.

[7] P. J. Landin. The Mechanical Evaluation of Expressions. *Computer Journal*, 6(4):308–320, 1964.

[8] B. Loo et al. Declarative Routing: Extensible Routing with Declarative Queries. In *Proceedings of SIGCOMM 2005*, pages 289–300.

[9] C. Partridge et al. RFC 1546 - Host Anycasting Service, November 1993.

[10] G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, University of Aarhus, 1981.

[11] J. H. Saltzer. Naming and Binding of Objects. In Rudolph Bayer et al. (eds.), *Operating Systems - An Advanced Course*, pages 99–208. Springer LNCS 60, 1978.

[12] I. Stoica et al. Internet Indirection Infrastructure. *IEEE/ACM Transactions on Networking*, 12(2):205–218, April 2004.

Protocol Design Beyond Graph-Based Models

Thomas Moscibroda
Microsoft Research
moscitho@microsoft.com

Roger Wattenhofer
ETH Zurich
wattenhofer@tik.ee.ethz.ch

Yves Weber
ETH Zurich
webery@tik.ee.ethz.ch

ABSTRACT

In this paper we shed new light on the fundamental gap between graph-based models used by protocol designers and fading channel models used by communication theorists in wireless networks. We experimentally demonstrate that graph-based models capture real-world phenomena inadequately. Consequentially, we advocate studying models beyond graphs even for protocol-design. In the main part of the paper we present an archetypal multi-hop situation. We show that the theoretical limits of any protocol which obeys the laws of graph-based models can be broken by a protocol explicitly defined for the physical model. Finally, we discuss possible applications, from data gathering to media access control.

1 INTRODUCTION

Wireless multi-hop networks such as sensor, ad hoc, or mesh networks are often modeled by means of graphs. In the most general model, two graphs are given: A connectivity graph $G_c = (V, E_c)$ and an interference graph $G_i = (V, E_i)$. Both graphs are based on the set of devices V. A receiver v successfully decodes a message from a sender u, if and only if u and v are neighbors in the connectivity graph, $(u, v) \in E_c$, and v does not have a concurrently transmitting neighbor node in the interference graph G_i. Protocol designers often consider special cases of this general model. For example, it is sometimes assumed that $G_i = G_c$, or that G_i is G_c augmented with all edges between 2-hop neighbors in G_c.[1] In graph-based models, a protocol designer has to take care that no neighbor of in G_i is transmitting simultaneously to a neighbor in G_c, or at least, that this happens rarely.

Graph-based models have been particularly popular with higher-layer protocol designers, as they abstract away real-world complications. On the other hand, the concept of an *edge* is oversimplifying starkly, as it is a binary representation for continuous (non-binary!) physicals laws. In fact, nodes barely outside the interference range of a receiver v (that is, a node not connected by an edge with v in G_i) might still cause enough cumulated interference such that receiver v is not able to decode a message from a legitimate neighboring sender in G_c.

Communication theorists on the other hand often do not employ graph-based models. Instead they are studying an arsenal of fading channel models, the simplest being the signal-to-interference-plus-noise ratio (SINR) model. In the SINR model, the energy of a signal fades with the distance to the power of the path-loss parameter α. If the signal strength received by a device divided by the interfering strength of competitor transmitters (plus the noise) is above some threshold β, the receiver can decode the message, otherwise it cannot. This simple SINR model is unrealistic as well, mostly because it is inherently geometric: In reality antennas are not perfectly isotropic, and even more importantly the environment is obstructed by walls or plants. Although these issues can be integrated into the basic SINR model, these "SINR+" models are predisposed to get complicated – essentially intractable from the point of view of a protocol designer. Graph-based models on the other hand automatically incorporate both imperfect (or even directional) antennas and terrains with obstructions. It seems that a majority of classes, books, or tutorials therefore prefers to teach higher-layer concepts in wireless multi-hop networking in terms of graphs, not in terms of SINR.

Even though SINR models allow for exciting scaling law studies (e.g. the theoretical capacity of wireless multi-hop networks), they are often too complicated to comprehend a protocol, let alone analytically prove correctness and/or efficiency of a protocol.

We believe that bridging the gap between protocol designers and communication theorists is a fundamental challenge of the coming years, a hot topic for the wireless multi-hop community with implications for both theory and practice. In particular, in this paper, we advocate studying models beyond graphs, especially for *protocol-design*. After some introductory back-of-the-envelope calculations in Section 2, Section 3 presents experimental results that show that even vanilla sensor radios with restricted hardware can achieve communication patterns which are impossible in graph-based models. In Section 4 we head beyond these straight-forward examples and fantasize about the applications of the experimental findings; in particular we present an archetypal multi-hop

[1] Alternatively, it is sometimes assumed that these graphs are the result of a geometric setting. In particular the nodes are points in the Euclidean plane. Two nodes are neighbors in G_c if their Euclidean distance is at most 1. Two nodes are neighbors in G_i if their Euclidean distance is at most r, for some parameter $r \geq 1$. This model is widely known as the *unit disk graph* model with interference.

situation where we propose routing/transport schemes which may break the theoretical throughput limits of any protocol which obeys the laws of a graph-based model. Sections 5 and 6 then discuss related work and future directions, respectively. Note that our examples are preliminary in the sense that they are geared towards illustrating basic concepts and highlighting the fundamental problems of graph-based modeling, rather than towards maximizing the achievable throughput.

2 MOTIVATING EXAMPLES

Consider a network of n devices $x_1, x_2, \ldots, x_n$. A message from a transmitter x_s can be correctly decoded by a receiver x_r if and only if $\frac{P_r}{I_r+N} \geq \beta$ for a hardware-dependant ratio β. In this equation, P_r is the signal strength of the message at the receiver, I_r is the sum of all interferences at x_r, and N is the ambient noise.

In the *physical model* of signal propagation [8] the signal strength P_r is modeled as a polynomially decreasing function depending of the distance $d(x_s,x_r)$ between the sender and the receiver. More precisely, it is assumed that $P_r = \frac{1}{d(x_s,x_r)^\alpha}$ where α is called the *path-loss exponent*, a constant dependent on the medium, typically between 2 and 6.

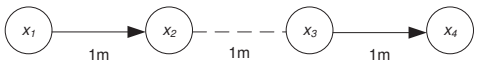


Figure 1: Four nodes placed equidistantly in a line.

Consider the simple "hidden-terminal" network with 4 nodes illustrated in Figure 1.[2] Nodes x_1 and x_3 want to send a message to the corresponding receivers x_2 and x_4, respectively. A graph-based communication model implies that at least *two* time slots are required. Otherwise, the two messages would collide at x_2.

In the physical SINR model, however, the two messages can be easily sent in parallel. For a simple calculation, assume $\alpha = 3$, $\beta = 3$, and background noise $N = 10$ nW. Those values are realistic, even pessimistic, in sensor networks [12] as well as other forms of wireless networks. Let $\beta_{x_i}(x_j)$ be the SINR ratio at a node x_i in which the signal power from node x_j is considered "signal" and the signal power of all other simultaneously transmitting nodes is considered interference. That is, a node x_i successfully receives a message from x_j if and only if $\beta_{x_i}(x_j) \geq \beta$.

If x_1 and x_3 send with power $P_{x_1} = 0$ dBm and $P_{x_3} = -7$ dBm, respectively, we get the following SINR values: $\beta_{x_2}(x_1) = \frac{1000\ \mu\text{W}/(1\text{ m})^3}{0.01\ \mu\text{W}+(200\ \mu\text{W}/(1\text{ m})^3)} \approx 5.00$ and $\beta_{x_4}(x_3) = \frac{200\ \mu\text{W}/(1\text{ m})^3}{0.01\ \mu\text{W}+(1000\ \mu\text{W}/(3\text{m})^3)} \approx 5.40$. Consequently, both receivers can correctly decode their corresponding message without any problems.

[2] Depending on the specific application scenario, all four nodes may be sensors in a wireless sensor network or stations in a wireless mesh network. Alternatively, nodes x_2 and x_4 may be base stations and x_1 and x_3 may be clients in a Wireless LAN.

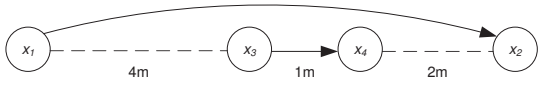


Figure 2: A more elaborate example with four nodes.

Now we consider a more elaborate example by rearranging the two sender-receiver pairs (x_1, x_2) and (x_3, x_4) in a way that one pair is placed in the transmission line of the other. This setup is shown in Figure 2. As before, the question is whether it is really necessary to schedule the two messages in succession or if they can be sent in the same time slot without colliding at any of the two receivers. Clearly, any graph-based approach trying to send the two messages in parallel will fail because, intuitively, the medium between x_3 and x_4 can only be used once per time slot.

In the SINR model, however, both messages can easily be transmitted simultaneously, thereby doubling the achieved throughput. When x_1 sends with $P_{x_1} = 1$ dBm and x_3 with $P_{x_3} = -15$ dBm, we get the signal-to-noise and interference ratios of $\beta_{x_2}(x_1) = \frac{1.26\text{ mW}/(7\text{ m})^3}{0.01\ \mu\text{W}+(31.6\ \mu\text{W}/(3\text{ m})^3)} \approx 3.11$ and $\beta_{x_4}(x_3) = \frac{31.6\ \mu\text{W}/(1\text{ m})^3}{0.01\ \mu\text{W}+(1.26\text{ mW}/(5\text{m})^3)} \approx 3.13$. That is, the SINR ratios are such that node x_4 can perfectly decode x_3's message, and at the same time, x_2 successfully receives x_1's message. There is no collision.

3 PROMISING EXPERIMENTS

After these theoretical considerations, this section shows that the effects described above are not only theoretical shenanigan, but can be verified with widely used standard sensor nodes. We decided to employ the mica2 sensor nodes running with TinyOS. They are equipped with a ChipCon CC1000 radio transceiver configured to send at a frequency of 868 MHz.

3.1 Two Pairs of Nodes

We created a testbed with two senders x_1 and x_3 and two corresponding receivers x_2 and x_4 positioned on a line similar to the setup shown in Figure 2. The distances between the nodes were scaled down to 100 cm, 30 cm, and 60 cm. The sender tries to transmit 20000 messages in succession to the corresponding receiver which counts the number of messages received.

For the success of this experiment, it was crucial that the MAC layer allows parallel transmission of multiple messages. Consequently, we adjusted the collision-preventing MAC layer delivered with TinyOS: Before sending a message, no check is performed if the medium

is free. Additionally, the initial random backoff before sending a message was removed. The output powers were fixed to 0 dBm for x_1 and -10 dBm for x_3. We refer to this adjusted protocol as "SINR-MAC".

Before presenting the measurement results, we calculate a theoretical lower bound for the time required to transmit the 20000 messages when assuming a graph-based communication model. A single packet containing 6 bytes of payload requires transmitting 23 bytes due to preamble, header, etc. The sensor sends with a data rate of 2.4 kBps and switching from RX to TX mode and back to RX mode requires about 0.5 ms, according to the CC1000 data sheet.[3] Summed up, at least $(23 \text{ bytes}/2.4 \text{ kBps} + 0.5 \text{ ms}) * 40000 \text{ packets}) \approx 403$ s are required when assuming a graph-based model. Note that this lower bound is very conservative, ignoring any software overheads.

We obtained a value of 603s for x_1 and 591s for x_3 using the standard TinyOS MAC layer protocol.[4] In contrast, the "SINR-MAC" only required 267s (x_1) and 268s (x_3), respectively. This performance gain did not have negative effects on the reliability: With the standard MAC layer, x_2 received 19998 messages and x_4 received 18852 messages. The corresponding values using the "SINR-MAC" are 18668 for x_2 and 19916 for x_4.

These results show that the examples analyzed in the previous section can be implemented in practice. On the one hand, the time used by the default MAC layer protocol exceeds the calculated lower bound by almost 50%. On the other hand, the "SINR-MAC" exploiting the interference phenomena of the SINR model performs significantly better than this limit. This highlights the inherent inability of graph-based models to represent important physical aspects that govern real sensor network. More importantly, however, this experiment shows that a protocol that is specifically tailored to the SINR model—in this case the adjusted "SINR-MAC" layer protocol—can outperform conventional, implicitly graph-based protocols by a factor of 2 or more.

3.2 Multiple Pairs of Nodes

Delighted by the results of the experiment above, the question arises if standard sensor nodes allow to use the medium threefoldly. The setup was analogous to the previous measurement with an additional sender and receiver pair, as shown in Figure 3. The output power of the senders was set to 0 dBm (x_1), -10 dBm (x_3), and -20 dBm (x_5).

The distances between the nodes were found by trial and error. During the search for promising distances,

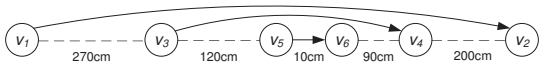


Figure 3: Three interleaved sender-receiver pairs.

we noticed that this setup is less failure tolerant than the first experiment with only two sender and receiver pairs: While moving a node a few centimeters to the left or to the right did not produce significant changes in the results, bigger changes led to complete failure of a receiver, i.e. it did no longer receive any messages destined for it. The reason is that in this experiment, each sender now has two competitors, whose interferences cumulate and reduce the region with sufficient SINR.

In this experiment, the same parameters as in the experiment above were measured, i.e. the time required by the three senders to completely send all 20000 messages and the number of successfully decoded messages.

	Time required	
	standard MAC	"SINR-MAC"
Node x_1	721 s	267 s
Node x_3	778 s	268 s
Node x_5	780 s	270 s

The number of successfully received messages at x_2, x_4, and x_6 using the standard MAC protocol was 19999, 18784, and 16519, respectively. For the "SINR-MAC", the corresponding values were 19773, 18488, and 19498.

These measurements further emphasize the inability of graph-based approaches to model real sensor networks. The time required to send the 20000 messages is invariant even with three nodes sending messages while the standard MAC layer—as graph-based calculations would suggest—requires almost three times longer. Additionally, the number of collisions increased for the default MAC layer protocol resulting in a packet loss rate of 7.83% while the adjusted MAC layer shows a more or less invariant rate compared to the previous results.

Building systems with four or more senders transmitting messages in parallel becomes more and more impractical. On one side, this is because each additional sender increases the interference at the other senders. On the other side, the radio module only supports a limited interval of output powers. Our experiments have shown that four senders placed in a line are possible under perfect conditions. However, such systems tend to be very failure-prone in real environments. But different hardware platforms may produce different results.

4 Applications & Challenges

In the previous sections, we have seen how graph-based models inherently fail in capturing certain important physical phenomena. The fact that graph-based models do not properly describe all aspects of physical reality

[3] ChipCon AS, SmartRF CC1000 Datasheet (rev. 2.2), `http://www.chipcon.com/files/CC1000_Data_Sheet_2_2.pdf`

[4] All presented results are from one single run; however, we repeated all tests, and obtained similar results.

is of course neither new nor surprising (see for instance [2, 7, 10]). The more interesting question is whether the resulting inconsistencies are small enough to be justified by the gained simplicity of the model. Moreover, it is important to ask whether physical phenomena that cannot be modeled as graphs can be exploited for designing (and analytically proving!) algorithms and protocols that break the theoretical boundaries placed by graph-based models. In other words, we raise the question whether there are applications for wireless multi-hop networks in which provably efficient (possibly even theoretically optimal) graph-based algorithms perform inherently worse than algorithms that are designed to make use of SINR aspects. If there were no such examples, it would serve as a major justification for studying networks on the clean abstraction layer of graphs. If, however, there are examples in which the performance of theoretically optimal graph-based algorithms is surpassed by algorithms that explicitly take SINR into account, it would highlight the need for a more physical-level oriented design and analysis of network protocols.

One important application is *data gathering* with high throughput requirements in *heterogeneous wireless multi-hop networks*. Specifically, consider a heterogeneous network with potentially energy-restricted wireless nodes that gather data and locally distribute or forward this data for aggregation, and a few designated, more powerful nodes. Eventually, the data has to be sent to a base station, a task which is preferably done by the long-range nodes, instead of the regular sensor nodes. In any graph-based model, local communication among regular nodes and long-range communication among designated nodes must be coordinated (either in the time or frequency space, or by using spatial multiplexing). As in the four-node example of Figure 2, however, long-range and short-range communication can *coexist*, that is, regular nodes can communicate with each other *while* long-range nodes send data to the base station. This could result not only in *higher throughput*, but also in a significantly *smaller coordination overhead* between different regions of the networks. Other applications could include improving the capacity in *wireless mesh networks* or even *cellular networks*.

In the remainder of this section, we want to theoretically study an application in which, even from an information-theoretic point of view, the theoretical limitations of graph-based models can be surpassed when explicitly using protocols designed for SINR environments.

Improving Channel-Throughput

Consider a *multi-hop channel* consisting of a chain of wireless nodes. The left-most node is the sender that wants to send data to the right-most node, its destination. Nodes being power constrained, the messages must be forwarded in a multi-hop fashion from source to destination. The question is, at what *rate R* can data be transmitted in this model, that is, how much information can be successfully transmitted from source to destination in a certain time-interval.

In the formal model, the chain consists of n equidistantly placed nodes $x_1, \ldots, x_n$, where x_1 and x_n are the source and destination, respectively. In the graph-based model, the *maximum transmission range* of any node is denoted by ℓ, $\ell < n$, i.e., a node x_i can send a message to $x_{i+\ell}$ in the absence of any interference.

We do not consider complex wireless signal propagation models because, interestingly, it suffices to study the basic *physical model* [8] in order to highlight the difference to graph-based models. Also, all our lower bounds hold even in very simplistic and optimistic graph models in which the interference range equals the transmission range, and in which time is divided into globally synchronized slots. Clearly, in more realistic graph models in which the interference range exceeds the transmission range, and in case of asynchrony, the achievable rates would be even worse.

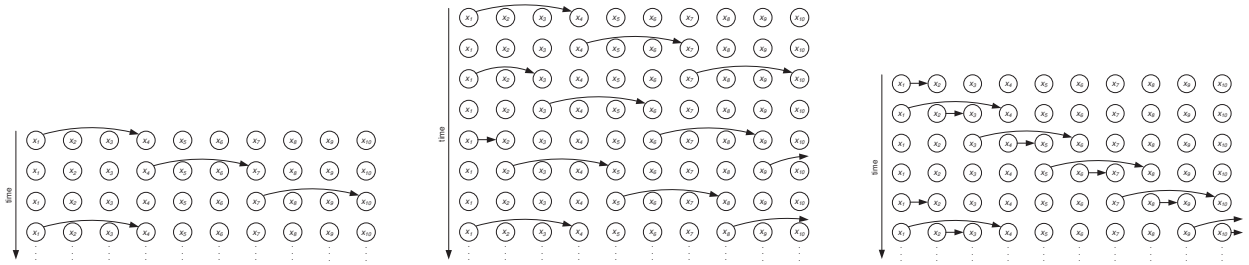
We begin by showing that the naive idea to ship information from x_1 to x_n achieves only a very moderate rate. Consider the protocol in which every node transmits at power ℓ', for some $1 \leq \ell' \leq \ell$. When having a message, a node x_i sends this message to $x_{i+\ell'}$ at the earliest opportunity. It can be seen in Figure 4(a) that x_1 can insert a message into the chain in time slot 1, but then has to wait for 2 consecutive time slots, before injecting its next message. The reason is that node $x_{\ell'+1}$ experiences interference during these two slots. As x_1 can thus insert a new message into the chain only once every three time slots, the achieved rate is $R = 1/3$.

Observation 1. *The rate achieved by the naive graph-based protocol of Figure 4(a) is* $R = 1/3$.

Clearly—even in graph-based models—much better protocols can be devised when using *power control*. Specifically, the rate can be improved by employing the forwarding scheme shown in Figure 4(b). Intuitively, if sending a message to its ℓ'-hop neighbor is impossible due to interference at the receiver, the message is forwarded to a closer neighbor where reception is possible. It can be shown that in this scheme, the channel allows the injection of 3 packets every 7 time slots.

Observation 2. *The rate achieved by the improved graph-based protocol of Figure 4(b) is* $R = 3/7$.

By using more complicated graph-based techniques, this rate may be improved further. However, the following theorem proves that even with the most sophisticated power control scheme and scheduling approach, the rate of $1/2$ can never be surpassed by a protocol that obeys the laws of a graph-based model.



(a) Naive graph-based (b) Clever graph-based (c) SINR-based

Figure 4: Figure 4(a) shows a naive, graph-based approach to send data from x_1 to x_n using $\ell' = 4$. A more sophisticated method to send messages from x_1 to x_n achieving a rate $R = 3/7$ is shown in Figure 4(b). The scheme in Figure 4(c) explicitly employs the SINR model to send messages from x_1 to x_n.

Theorem 3. *The maximum achievable rate R of graph-based scheduling protocol is*

$$R \;\leq\; \frac{1}{2} - \frac{1 - \frac{2\ell^2}{n}}{4(\ell + \frac{1}{2})}.$$

For $\ell < \sqrt{2n}/2$, this is strictly smaller than $1/2$.

Proof. Consider an arbitrary time slot. If in this time slot a node x_i transmits a message to node x_j that is h_i hops away ($j = i + h_i$), there cannot be any message sent to a node in the interval $x_{i-h_i}, \ldots, x_i, \ldots, x_{i+h_i}$. We call such nodes *blocked*, because no message can be sent to them. Let h_i be the number of hops node x_i transmits its message in this time slot, and let b_i denote the resulting number of blocked nodes. The following relation holds between h_i and b_i: $b_i = 2h_i + 1$ for $i \geq 2 + h_i$ and $b_i \geq h_i + 1$ for $i < 2 + h_i$.

Notice that there can be at most one successful sender x_i with $i < 2 + h_i$ in one time slot. Let S be the set of indices of successful senders in this time slot and let $P := \sum_{i \in S} h_i$ denote the amount of *progress* achieved by all nodes in the chain. The number of blocked nodes can be at most n, which implies that

$$i' + h' + 2 \cdot \sum_{i \in S \setminus \{x_{i'}\}} h_i + |S \setminus \{x_{i'}\}| \;\leq\; n, \qquad (1)$$

where h' and i' are the hop-distance and the location of the left-most transmission, respectively. Because $i' \geq 1$ and $h' \leq \ell$, this can be rewritten as $\sum_{i \in S} h_i \;\leq\; \frac{1}{2}(n + \ell - |S|)$. On the other hand, it is clear that $\sum_{i \in S} h_i \leq \ell \cdot |S|$ holds because every node in S can at most send over ℓ hops. Hence, the progress P in every time slot is bounded by $P \;=\; \sum_{i \in S} h_i \;\leq\; \min\left\{\frac{1}{2}(n + \ell - |S|), \ell|S|\right\}$, which is maximized when $|S| = \frac{n+\ell}{2(\ell+1/2)}$. Plugging in this value, the resulting progress is at most

$$P \;\leq\; \frac{n}{2} - \frac{n - 2\ell^2}{4(\ell + 1/2)}.$$

The theorem now follows because an algorithm with rate Λ must achieve a total progress of at least $tn \cdot \Lambda$ in t time slots, when $t \to \infty$. Because progress in each time slot is bounded by P, however, the achievable rate is at most P/n, which yields the claimed result of the theorem. □

In view of this theorem, the question is whether $1/2$ is a fundamental barrier that cannot be surpassed by *any* protocol, or whether it is imposed solely by the underlying graph model. As it turns out, the latter is the case and depending on the values of α and β, the achievable rate can be at least $1/2$. In the scheme illustrated in Figure 4(c), for instance, x_1 first sends a packet to its one-hop neighbor. In the second iteration, this packet is forwarded one additional hop, to x_3. Simultaneously —and this is where we abandon the graph-based model—the sender x_1 transmits a second packet to x_4. As shown in Section 2, this is possible in the SINR model. Subsequently, these two messages are forwarded in the same manner in every time slot: the trailing message "hops" over the leading message, until they reach the destination. When doing the necessary calculations in the SINR model, it can be shown that for some α and β and appropriate power levels, this scheme can reach a rate of $R = 1/2$, because x_1 can inject a new packet in two time slots out of four. In fact, it may be the case that more sophisticated SINR-based schemes than the one shown in Figure 4(c) reach a rate strictly larger than $1/2$.

Observation 4. *For certain values of α and β, SINR-based scheduling protocols achieve a rate of $R \geq 1/2$.*

Notice that in its current form, the scheduling protocol of Figure 4(c) is valid only for large α and small β, and it may not be practically employable in certain settings for this reason. However, it serves to illustrate the potential gain in throughput by employing protocols and algorithms explicitly and making use of SINR phenomena. In particular, Observation 4 shows that by using the method of consecutively "overtaking" messages, the achievable rate can be $1/2$, whereas Theorem 3 proves that no graph-based protocol can achieve such a rate.

5 Related Work

The discrepancy between graph-based models and physical SINR models has been recognized by researchers many years ago. For instance, the papers [2, 7] evaluate the performance of graph-based scheduling protocols in SINR environments by means of simulations and on the

assumption that nodes are distributed uniformly in the plane. Our work goes beyond these papers in the sense that we suggest to actually *design protocols* explicitly for SINR-based models, thus improving currently employed protocols. In fact, when it comes to scheduling, there already exist numerous algorithms in SINR environments, including for instance [4–6]. The authors of [4, 5] study the problem of finding a schedule and power control policy that minimizes the total average transmission power in the wireless multi-hop network. While these works provide important results, the proposed algorithms do either not yield efficient guarantees in arbitrary networks or are based on solutions to complex non-polynomial optimization problems. Computationally efficient solutions with provable guarantees that utilize SINR effects similar to the ones in this paper have only recently been studied for scheduling [10] and topology control [11].

In systems research, the general idea of exploiting "collision-but-not-failure" effects has been considered by Whitehouse et al., which makes explicit use of the *capture effect* [13]. Our work is different in that nodes actively select their power levels appropriate for creating desirable capture effects.

There has recently been a tremendous research effort towards increasing throughput in wireless networks, and some proposed strategies go beyond graph-based models [3, 9]. Biswas and Morris propose an improved routing and MAC layer protocol to enhance throughput in wireless networks [3]. Katti et al. propose an architecture for wireless mesh networks that disposes of the point-to-point abstraction of wireless channels and is based on the idea of *network coding* [9]. Since neither paper exploits SINR-effects at the *physical reception* of messages, they abandon the graph-based model on a different layer than we do. That is, applying ideas from network coding is completely *orthogonal* to our proposal, and hence can be applied in combination with SINR-based methods to further improve the results.

Another direction towards improving network capacity has been to use multiple or cognitive radios and allow communication on different frequencies, e.g. [1]. Again, these strategies are orthogonal to our work because SINR-effects can be exploited at each frequency individually.

6 Conclusions & Outlook

Sections 3 and 4 have shown that protocols explicitly designed with SINR in mind can surpass the theoretically achievable performance of any graph-based protocol even in simple settings. The real challenge of course is to take these observations one step further, both theoretically and practically. From a *theoretical* point of view, it would be interesting to further characterize the gap between achievable rates in networks based on graph models, the physical model, or even more realistic signal propagation models. While there is ample work dealing with exactly these kind of questions, we believe that there is still a lack of fundamental *algorithmic foundation* that allows to transform the theoretical insights into efficient and practical network protocols.

Even more important challenges, however, arise from *practical aspects* of our observations, i.e., turning them into practical network protocols. It would for instance be intriguing to study whether MAC layer protocols with higher throughput could be devised. Also, there is a large potential for improving the throughput of routing protocols or data gathering applications by incorporating our ideas. The ultimate challenge will be to circumventing the inevitably arising practical difficulties in order to tap the full potential of these technologies.

References

[1] P. Bahl, R. Chandra, and J. Dunagan. SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks. In *Proc. of the 10th Int. Conference on Mobile Computing and Networking (MOBICOM)*, 2004.

[2] A. Behzad and I. Rubin. On the Performance of Graph-based Scheduling Algorithms for Packet Radio Networks. In *Proc. of the IEEE GLOBECOM*, pages 3432–3436, 2003.

[3] S. Biswas and R. Morris. ExOR: Opportunistic Multi-Hop Routing for Wireless Networks. In *Proc. of ACM SIGCOMM*, Philadelphia, Pennsylvania, USA, 2005.

[4] R. L. Cruz and A. V. Santhanam. Optimal Routing, Link Scheduling and Power Control in Multi-hop Wireless Networks. In *Proc. 22th IEEE INFOCOM*, 2003.

[5] T. ElBatt and A. Ephremides. Joint Scheduling and Power Control for Wireless Ad-hoc Networks. In *Proceedings of the 21th IEEE INFOCOM*, 2002.

[6] J. Grönkvist. Assignment Strategies for Spatial Reuse TDMA. *Licentiate thesis, Royal Institute of Technology, Stockholm, Sweden*, 2002.

[7] J. Grönkvist and A. Hansson. Comparison Between Graph-Based and Interference-Based STDMA Scheduling. In *Proc. 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MOBIHOC)*, pages 255–258, 2001.

[8] P. Gupta and P. R. Kumar. The Capacity of Wireless Networks. *IEEE Trans. Information Theory*, 46(2):388–404, 2000.

[9] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. XOR in The Air: Practical Wireless Network Coding. In *Proc. of ACM SIGCOMM*, Pisa, Italy, 2006.

[10] T. Moscibroda and R. Wattenhofer. The Complexity of Connectivity in Wireless Networks. In *Proc. of the 25th IEEE INFOCOM*, 2006.

[11] T. Moscibroda, R. Wattenhofer, and A. Zollinger. Topology Control meets SINR: The Scheduling Complexity of Arbitrary Topologies. In *Proc. of the 7th ACM Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2006.

[12] D. Son, B. Krishnamachari, and J. Heidemann. Experimental Analysis of Concurrent Packet Transmissions in Low-PowerWireless Networks. Technical report, Viterbi School of Engineering, University of Southern California, 2005.

[13] K. Whitehouse, K. Woo, F. Jiang, J. Polastre, and D. Culler. Exploiting the Capture Effect for Collision Detection and Recovery. In *Proc. of the 2nd IEEE Workshop on Embedded Networked Sensors (EmNetS-II)*, pages 45–52, 2005.

Some Implications of Low Power Wireless to IP Networking

Kannan Srinivasan[†], Prabal Dutta[‡], Arsalan Tavakoli[‡], and Philip Levis[†]
[†]Computer Systems Laboratory, Stanford University, Stanford, CA
[‡]Computer Science Division, UC Berkeley, Berkeley, CA

Abstract

We examine and outline challenges in IPv6 routing over low-power wireless personal area networks (PANs). We present empirical measurements and analysis of an increasingly popular PAN link layer, 802.15.4. We show that over short periods 802.15.4 exhibits bimodal connectivity, but over longer periods has many intermediate links. We quantify how synchonous acknowledgments affect common low-power routing metrics, such as ETX. We identify metrics for detecting modal changes in link quality. We explore how these behaviors affect IP routing and IPv6 requirements, such as route selection and maintenance, sub-IP fragmentation and assembly, and packet scheduling.

1 Introduction

Low-power wireless is increasingly important to computer networking. As Moore's Law has pushed the price and form factor of computers down, networks have expanded to include large numbers of wireless desktops, laptops, palmtops and cellphones. This trend towards smaller, lower power, and more numerous devices has led to new wireless physical and data-link standards to support them, such as Bluetooth [4], 802.15.4 and 802.15.4b [18], which are designed for short range personal area networks (PANs). Economies of scale may make PAN devices more numerous than any other class of networked node. In order to maximize lifetime, PAN devices aggressively conserve energy.

Wireless sensor networks (sensornets) are one heavily studied subclass of PANs [8]. Composed of collections of tiny, battery- limited devices with a few kB of RAM, a few MHz of CPU, and sub-1% duty cycles, sensornets impose novel and unique network requirements. Research sensornet architectures [7, 10] as well as industrial standards [1] have discarded IP, arguing that it is not suitable due to addressing, network dynamics, discovery, and power. Instead, research protocols have focused on data-centric approaches, while standards such as Zigbee have defined monolithic stacks that stretch from the data-link to the application layer.

Not everyone agrees that IP is inappropriate. The IETF has recently formed a working group – 6lowpan – to define how to run IPv6 on low-power PAN protocols [12]. 6lowpan believes that the expected number of devices calls for an enormous address space, making IPv6 better suited than IPv4. There are many reasons why IP is attractive, including interoperability, a huge library of tools and utilities, and decades of research towards understanding its behavior. History has shown IP to be flexible enough for many different networks and usage patterns, working well, or at least well enough, in many domains for which it was never initially intended.

This debate raises two closely related questions. First, how do low-power wireless networks behave? Second, what implications do these behaviors have for IP? The first question has been an important area of sensornet research. Several studies have experimentally quantified low-power wireless radio performance and behavior by exploring the effects of environments, encoding, frequencies, and by disambiguating causes of loss [5, 6, 9]. At this point it is clear that low-power wireless has many differences from the media traditionally considered when discussing IP networking, such as bandwidth utilization, energy minimization, and packet sizes. As much of academia and research has dismissed IP, however, there has been little thought or investigation into the question of how these results would affect IP-based networking. Quantifying *how* low-power wireless is different and the correspoding implications is an important first step towards understanding the challenges in bringing IP to these devices.

This paper presents measurements of the long- and short-term behavior of the dominant PAN layer 2 protocol, 802.15.4. It shows ways in which it differs significantly from higher-power protocols in the same spectrum (e.g., 802.11b) as well as the low-power radios measured in early PAN/sensornet studies. It presents some implications of these behaviors to IPv6 networking. Table 1 summarizes the contributions of this paper as its experimental observations and their implications.

2 Background

The IEEE 802.15 working group focuses on wireless PAN protocols. More recently, the 802.15.4 task group was chartered "to investigate a low data rate solution with multi-month to multi-year battery life and very low complexity." 802.15.4 uses periodic beacons that conserve energy by scheduling communication without requiring an association protocol. 802.15.4 uses CSMA for media access. In terms of raw bandwidth per joule, 802.11b is cheaper; what makes 802.15.4 attractive to PANs is its simpler electronics, which lead to lower cost, faster wakeup, and lower sleep currents.

Two aspects of the 802.15.4 MAC layer are particularly important to IPv6 networking. The first is synchronous layer 2 acknowledgments. When a node sends a unicast packet, it can request an acknowledgment from the receiver, which the receiver sends approximately 180μs later. An acknowledgment packet is 5 bytes long, containing only the format header (2 bytes), a CRC (2 bytes), and the received packet sequence number (1 byte): it contains neither a source nor a destination address. The second is that the maximum 802.15.4 packet size is 127 bytes. The 128th byte is used by the physical layer to denote the size of the packet. This is important because IPv6 requires data-link layers whose MTU is smaller than 1280 octets to provide a sub-IP fragmentation and assembly layer. The expectation is that few PAN packets will be large, but this functionality is a requirement for IPv6 interop-

Observation	Section	Implications
Over short packet bursts, links qualities are largely bimodal.	Sec. 3	Fragments may be sent in small bursts when a link is good (greedy link select). Need sub-IP acknowledgment scheme to handle fragment flushes.
Low rate traffic encounters intermediate links, which are due to SNR variations or proximity to the reception threshold.	Sec. 4	Routing low utilization traffic requires continuous link estimation or route probing/discovery. The network layer may benefit from physical-layer information such as signal strength and noise measurements.
ETX asymmetries exist and are more common in low rate than burst traffic.	Sec. 5.	Route discovery cannot assume bidirectional communication. Routes require periodic refreshing or probing.
Packet ACK failures are correlated.	Sec. 5	Naive retransmissions waste energy. Need feedback between retransmissions and route selection. Need retransmission and duplicate suppression techniques.

Table 1. Summary of observations and their implications to IPv6 routing.

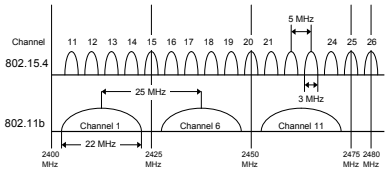
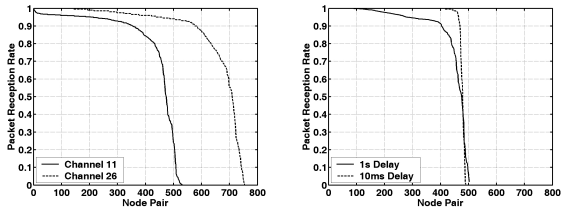


Figure 1. 802.11b and 802.15.4 spectrum utilization.



(a) Low-rate round-robin traffic.

(b) Short packet bursts on channel 26.

Figure 2. PRR distributions for a 28 node indoor testbed where nodes are on the ceiling. Reception rates are generally bimodal, and the commonality of intermediate links increases with inter-packet delays.

erability, and the 6lowpan working group has proposed an approach which incorporates header compression and the ability to use short 16-bit node addresses [14]. As PAN devices are energy-constrained, using techniques to increase single-hop delivery rates are valuable, as they can significantly improve end-to-end reliability and therefore reduce the number of network-level retransmissions.

The commonly used 802.15.4 physical layer occupies the same 2.4GHz spectrum as 802.11b. Because of their different data rates, their channels occupy different spectrum widths. Figure 1 shows their overlap and how 802.15.4 networks can experience interference from 802.11 networks. Unlike 15.4, which is assumed to have low utilization, a copresent 802.11 network might be very busy. While 802.11 might experience interference from 15.4, there is a 100-fold difference in output power: 802.11 chipsets have an output power of 15-23dBm, while 15.4 chipsets are typically -3-0dBm. The disparity makes it unlikely that an 802.11 node will act as a hidden terminal, as it requires the signal strength at the transmitter be below the clear channel threshold and be strong enough at the receiver to corrupt the packet.

However, as 802.15.4 has such a lower output power and is a narrowband interferer, 802.11 networks are not likely to respond to their transmissions when performing CSMA. 802.11 packets can be much briefer than 15.4 packets: a 300 byte packet at 11Mbps is approximately 200μs, while a 30 byte packet at 256kbps is 1ms. While most 802.11 data packets are 1500 bytes, acknowledgements and other control traffic often has smaller payloads. Therefore, a 15.4 node can detect a clear channel, start sending a packet, and receive a corrupting burst of mid-packet 802.11 interference.

2.1 Related Work

Experiments with early sensor platforms established that low-power wireless networks have complex dynamics. Ganesan et al [9] analyzed different protocol layers for rene motes, showing that simple algorithms such as flooding have significant complexity at scale. They observed that many node pairs had asymmetric packet reception rates (PRRs), which they hypothesized were due to reception sensitivity differences. Cerpa et al. [5] supported this hypothesis after swapping asymmetric node pairs and finding the asymmetries were a product of the nodes and not the environment. While the affects of link asymmetry have been studied in TCP traffic [3] and are applicable to PANs, the small packet sizes and temporal link variations raise separate issues, which to the best of our knowledge have not yet been addressed.

Cerpal et al. showed that PRR rates can change significantly over time, so that long-term PRR calculation can lead to very inaccurate results [6]. They suggested instead that an instanteous measure of RNP – "required number of packets" – was preferable to a long-term PRR. This work also introduced using conditional probabilities in link estimation, an idea which we extend when considering the correlation between packet failures in Section 5.

Aguayo et al. [2] observed similar packet delivery behaviors in a 38-node 802.11 long haul urban mesh network, but concluded that they were most likely due to multipath effects as there was little correlation between PRR and signal to interference plus noise ratio (SINR). Their experimental methodology differs from those of the sensor network studies. For example, they consider average SINR ratios over second-long periods rather than on a per-packet basis. Nevertheless, the differences in conclusions between the efforts are interesting. Since 802.11b operates in the same ISM band as 802.15.4 and uses a similar modulation scheme (QPSK), its transmitters could be significant interferers [19].

3 Distribution of Packet Reception Rates

Figures 2(a) and 2(b) show PRR disributions in an indoor testbed. Each point corresponds to a single, unidirectional

link over which at least one packet was delivered. There were a total of 28 nodes, giving 756 potential links. The figures specify the number of packets over which the reception rate is computed. Reception rates are sorted in descending order and the data for each line comes from a different experiment: each y-value for a given x-value is not expected to be from the same node pair. In the testbed, nodes were pinned to the ceiling, people moved freely through the space and there were 802.11 access points.

We measured PRR by having each node transmit 200 broadcasts under two different traffic patterns. In the first, *round-robin*, each node took turns transmitting a single packet, and transmissions were 500 ms apart. With 28 nodes, the inter-packet time for each node was 14 seconds, and for 200 packets the entire experiment took 47 minutes. In the second, *burst*, each node transmitted its 200 packets without interruption. With inter-packet times of 10 and 100 ms, the experiments took 56 seconds and 9 minutes.

Figure 2(a) shows that Channel 11 has 40% fewer high quality ($>$ 90% PRR) links than Channel 26. There are at least three possible explanations. First, 802.15.4 channel 11 shares spectrum with 802.11b while 802.15.4 channel 26 does not. Therefore, 802.11 traffic may interfere with 802.15.4 traffic on channel 11. This explanation, however, is not entirely satisfying. It seems that channel 11 should have a longer right tail since at least a few packets might have been received during the 47 minutes experiment on the 200 or more links seen on channel 26 but absent on channel 11. Second, since our experiments were carried out at different times, it is possible that the RF environment changed appreciably between the two trials. However, this explanation appears unlikely since repeating experiment at different times results in essentially the same distribution of reception rates versus node pairs. Third, the RF circuitry combined with the antenna on the mote may greater attenuate signals on channel 11 than channel 26. This, if true, can increase the communication range of a node and thus increase the number of neighbors for a node in channel 26.

Despite the absolute differences in Figure 2(a)'s distributions, both channels exhibit similar numbers of intermediate links (10%-90% PRR), approximately 150. Over the timeframe of about an hour, approximately 20% to 40% of the links had intermediate PRRs. In contrast, Figure 2(b) shows that over the much shorter timeframe of one minute (10 ms delay), packet reception is sharply bimodal. Approximately 85% of links exhibit a 100% reception rate, 10% of links have between 90% and 99% reception, while fewer than 5% of the links have a reception rate below 90%. When the timescale is increased to just over nine minutes (1 s delay), fewer than 20% of the links exhibit a 100% reception rate, 60% of the links exhibit between 90% and 99% reception rate, and 20% have a PRR below 90%.

Overall, the data indicate that distribution of PRRs in our indoor testbed are largely bimodal. The vast majority of links exhibit either greater than 90% or zero reception rate over short periods of time. The fraction of intermediate links over these timeframes is also small, as indicated by the pronounced knee and sharp fall-off in reception rate shown in Figures 2(a) and 2(b). These results show that over very short time scales, links are highly bimodal. As the time scale increases, the chances of link qualities changing increases, leading to larger but still small proportion of intermediate links. These observations contrast with the work of Aguayo et al. [2] which showed that in Roofnet – an outdoor 802.11b mesh network – between 50% to 70% of links have intermediate PRRs over a 90 second interval.

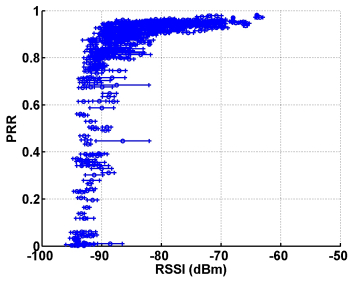


Figure 3. Packet Reception vs. Received Signal Strength for a long-term trace.

Our experiments did not include concurrent transmitters.[1] In the presence of hidden terminals, concurrent transmissions can lower packet reception rate due to collisions at the receiver. However, Aguayo et al. concluded that in their experiments, it seemed unlikely that interfering traffic caused the observed losses [2], so we can factor out foreign traffic as a source of significant differences in both cases. Even with 802.11b-induced interference, the distributions from the two experiments are considerably different.

Once a node detects a good link, that link is likely to be good for a burst of packets, such as a large IPv6 datagram. If a node has only a single burst to send, as soon as it finds a good link, greedily choosing that link may be a good strategy. In 802.15.4, a "good" link can still have a 5% packet loss rate. Succesfully transmitting a large IPv6 packet (10 fragments) therefore requires a sub-IP acknowledgment layer. Link-layer acknowledgments can provide one part of this mechanism, if a system follows the 6lowpan requirement that an overlapping fragment flush all other fragments, then imperfect duplicate suppression may cause a receiver to flush fragments that were acknowledged at the data link layer.

4 Intermediate Links

The previous section showed that 802.15.4 connectivity is highly bimodal and that the proportion of intermediate links increases over time. This section explores the reasons behind those observations and the implications to IP routing.

Prior studies established that signal-to-noise ratio (SNR) is the main factor determining packet reception success in low-power wireless [16, 17]. To verify this, we ran long term round-robin (8+ hours) experimental traces across different platforms, in varying environments. The results from all of these experiments were similar to those shown in Figure 3. When the RSSI is greater than some lower bound (-87dBm in this particular experiment), the PRR is high (greater than 80%) with a high likelihood. Otherwise the link falls into a grey area where the PRR is difficult to predict.

[1] Therefore our results may exhibit *fewer* intermediate links than a network with concurrent transmitters would.

SSI (dBm)	-98	-97	-96	-95	-94	-93	-92
# Nodes	5	8	4	3	2	3	1

Table 2. Distribution of the mode of noise readings across 26 802.15.4 nodes.

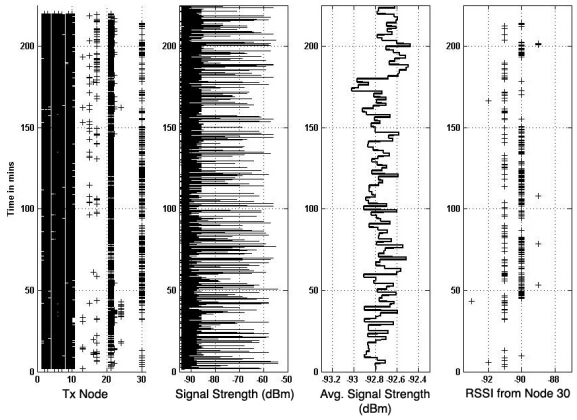


Figure 4. Observed behavior at a single node during a round-robin packet trace. The left plot shows packet delivery. The second plot shows SSI, which has many very brief spikes. The third plot shows the average SSI over 400 samples (40s) indicating that there are not significant long-term variations. The last plot, on the right, shows the RSSI distribution of packets received over time. Changes in PRR are correlated with RSSI variations.

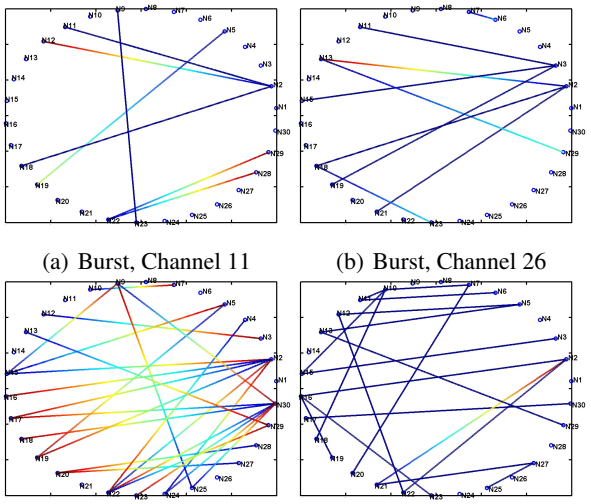
Table 2 shows the varied range of noise floors calculated as the mode of samples of the signal strength indicator (SSI). Note that SSI is not same as RSSI. RSSI is the signal strength of successfully received packets while SSI is the periodically sampled signal strength of the environment (noise). For the same RSSI, different nodes will see different PRRs due to differences in their noise floors.

After further investigation, we observed that not only do the unstable links have average SNRs that are on the edge of the "good link" threshold, but that the RSSI value of packets received from the same node can fluctuate by a few dBm over longer periods, as has been observed in other studies [13].

Figure 4 shows packet reception, noise, and RSSI data for a single node over a round-robin trace. The left graph shows packet reception over time for a single node (node 4) from all the other nodes in the experiment. During high PRR periods, the RSSI of the packets received from node 30 is predominantly -90dBm. The RSSI of received packets during poor periods s predominantly -91dBm or -92dBm. This slight drop in the received signal strength corresponds with a drop in the PRR. This shows how nodes whose SNR is in the edge of receive sensitivity experience temporal variations.

While these observations shed some light on the behavior of intermediate links, generalizing them to all environments is inappropriate. In these traces, for example, there were little correlation between noise spikes from 802.11b and PRR. This is possibly due to low traffic in the 802.11 network. However, if an 802.11 network is very busy it can cause packet losses, especially as 802.11 might not consider 15.4 traffic a busy channel. This can also lead to long term intermediate links.

This hypothesis suggests ways to identify intermediate



(a) Burst, Channel 11 (b) Burst, Channel 26

(c) Round-Robin, Channel 11 (d) Round-Robin, Channel 26

Figure 5. ETX asymmetries in burst and round-robin traffic. The nodes are in a circle solely for visualization purposes. Nodes close on the circle were physically close. Asymmetries a colored line, where the red end of the line is the node that had a higher ETX. A larger gradient indicates higher asymmetry.

links based on physical layer information (RSSI and noise floor). An IPv6 router, after identifying possibly intermediate links, may discard them to avoid unstable or time-varying routes. However, avoiding such links may result in a sparsely connected network and cause bottlenecks. Furthermore, discarding these links prevents greedy link selection that can make use of them during periods of good quality. Determining whether a borderline link is good at a particular time requires either periodic link maintenance or explicit probing. The results in Section 3 suggest that once a link is discovered to be good, it is likely to be good for a packet bursts

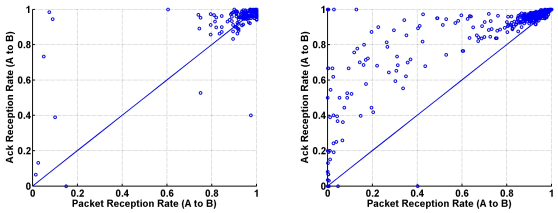
.

5 Acknowledgements

In this section, we examine the performance of 802.15.4 acknowledgments and how they affect link quality estimates. PAN devices often have limited RAM in order to minimize cost and energy consumption. This constraint makes 802.15.4's synchronous acknowledgments very valuable, as they have a bounded latency and so define how long a retransmission layer must hold onto a packet. However, losing link layer acks (false negatives) leads to unnecessary retransmissions and duplication of a packet within the network. Packet duplicates in turn require duplicate suppression techniques, which can increase the complexity of higher layers.

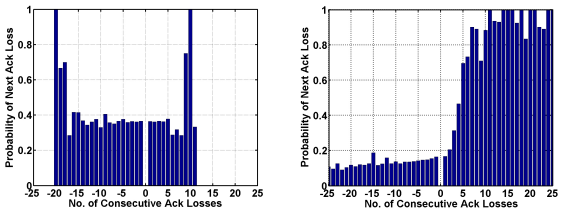
Existing energy-based route selection metrics such as ETX (the expected transmission count including retranmissions[20]) and its derivatives [11] use the product of forward and reverse packet reception rates. This assumes that the acknowledgment loss rate is the same as the packet loss rate in the reverse direction.

If the acknowledgment reception rate (ARR) can differ significantly from the reverse PRR, then it is possible that the two directions of a link have different ETX values, as the



(a) PRR vs ARR, Channel 11 Burst (b) PRR vs ARR, Channel 11 Round-robin

Figure 6. PRR for A→B link vs ARR for A→B. Burst traffic shows bimodal reception rates, while round-robin traffic shows more intermediate links. In almost all cases, ARR is higher than PRR. ARR and PRR are close at low loss rates, leading to few ETX asymmetries. Similar plots observed for channel 26 and are not shown for brevity.



(a) $A \rightarrow B$, Channel 11 (b) $B \rightarrow A$, Channel 11

Figure 7. Conditional probability of a packet not being acknowledged given n consecutive prior failures. The motes sent 100000 packets seperated by 10 msecs to each other in a burst. Negative numbers indicate n consecutive delivery successes. Acknowledgment packet losses are not independent. Channel 26 shows similar behavior but is not shown for brevity.

ETX from A to B (ETX_{AB}) is $\frac{1}{PRR_{AB} \cdot ARR_{BA}}$. There are two reasons why ARR may differ from PRR. First, 802.15.4 acknowledgment packets are very small, so are less likely to be corrupted. Second, CSMA causes a data packet transmission to suppress other nodes around it. As acknowledgments are shortly (tens of microseconds) after the data packet, the channel conditions around a transmitter are different than those at an arbitrary receiver.

For the purpose of this study, a link has an ETX asymmetry if the ETX for the two directions differs by 0.1 and at least one direction has an ETX below 3. The second condition is based on the observation that protocols typically minimize ETX. Figure 5 plots ETX asymmetries for burst and round-robin traffic on channels 11 and 26. Burst traffic on channel 11 observes 7 links with an ETX asymmetry, some of which are very asymmetric (N22-N28, N22-N29) while on channel 26 there are 9 asymmetric links, only one of which is very asymmetric (N2-N13). Round-robin traffic has many more asymmetries. On channel 11 most asymmetries are severe, while on channel 26 they are mostly slight.

Figure 5 shows that significant ETX asymmetries can exist, they are more pronounced over low-rate than bursty traffic, and channel choice affects the severity. As ETX asymmetries exist, ARR and PRR must differ. Figure 6 shows the relationship between PRR and ARR. As burst traffic observes predominantly bimodal links, its values are clustered at high reception rates. In contrast, round-robin traffic has more intermediate links. In both cases, however, the ARR is almost always greater than the PRR. Using PRR instead of ARR (as is commonly done in current protocols) overestimates ETX.

In Section 3, we showed that over long time periods links can have intermediate PRRs due to transitions between high and low short-term loss rates. An IP routing layer can easily handle either common case. The difficult case is when a link transitions in the middle of a packet or stream of packets.

Figure 7 shows what transitions look like to a routing layer. It shows the *conditional* probabilities of a successful data transmission and acknowledgement based on prior packets. This plot was generated from 100,000 transmissions between a single node pair with an intermediate loss rate. If failures are independent, then loss probabilities will be constant. Figure 7 shows conditional deliveries for each direction of a single node pair. Figure 7(a) shows failures that follow this pattern. The two edges of 100% loss represent rare cases. For example, there were 0 cases of 10, 3 cases of 11, and one case of 12, leading to values of 100% and 33%. Figure 7(b) shows a very different pattern, where packet losses are not independent: there are two cases, of approximately 10% loss and 80% loss. If a node B does not hear acknowledgments from A for several consecutive packets, then the probability of hearing future acknowledgments (whether due to data or ack failure) drops significantly.

The traces from the two experiments show a significant difference which explains these distributions. Approximately halfway through the burst from B to A, packet RSSI values increased for a long period, reaching an average of 5 dBm higher. This increase in RSSI similarly increased the packet delivery rate. The link underwent an RSSI shift, which transitioned it from the low quality to the high quality mode, producing an intermediate link. This is in contrast to the link from A to B, which during its burst happened to be on exactly the edge of receive sensitivity.

Link-level asymmetries preclude broadcast-based route selection techniques, such as those used in AODV [15]. Similarly, ETX asymmetries mean that the two directions of an IP route may differ. Just as with link quality variations, ETX asymmetries increase with time duration, and so routes require periodic probing or refreshing. As acknowledgments are imperfect and energy conservation generally calls for link-level retransmissions to improve reliability, nodes require duplicate suppression mechanisms. Packet loss correlation suggests that the sub-IP retransmission layer can provide useful feedback to IP route selection, telling it that a link has failed and choosing a different one will save energy.

6 Implications

Our experiments have four major observations.

1. Links are predominantly bimodal for short packet bursts.
2. Sporadic traffic observes intermediate links, which are due to SNR variations.
3. There are ETX asymmetries, which are larger over longer time intervals.
4. Acknowledgement failures are correlated.

The first and second observations indicate that once a node detects a good link, it should send IPv6 fragments as quick bursts. The bimodal delivery behavior means that there will be few reassembly failures at the receiver. However, as a link may transition from a good to a bad link during a transmission, a sender needs to maintain all fragments until a single recipient acknowledges all of the fragments.

The second and third observations together indicate that small, sproadic IPv6 packets and traffic bursts require different routing approaches, as link qualities may have changed. Continuously probing links (e.g., DSDV) or establishing routes (e.g., AODV) can easily consume more energy than data transmission. For latency-sensitive PANs, such as a lighting control system, this cost may be unavoidable. For less stringent PANs, however, such as a lawn monitoring or heating system, nodes can amortize route discovery costs by buffering packets into bursts. Alternatively, with physical layer knowledge a router can choose links with strong signal strengths, which are less likely to have temporal variations. As a single packet is sufficient for detecting a change in signal strength, this is an inexpensive measurement.

The third observation indicates that the two directions of an IP route may need to differ. The first observation implies that if the route is needed for a longer period then periodic rediscoveries may be needed, introducing a tradeoff in the cost of discovery and a route's energy efficiency. Continuously maintaining a routing table (e.g., DSDV) is also problematic, but the first observation implies that the rate at which the bidirectional quality of links need to be probed may consume a lot of energy. A novel routing protocol may combine parts of AODV and DSDV to overcome these challenges. A DSDV-like approach generates a set of candidate links, which are then probed with unicast messages to establish a route using an AODV-like approach, using seperate route requests may be needed for forward and back routes.

The fourth observation indicates that except for the few links which happen to be just at the reception sensitivity threshold, acknowledgments are an effective feedback mechanism for higher-layer decisions. A naive retransmission scheme will waste energy when there are several consecutive failures. A more sophisticated scheme that has an estimate of the cost-benefit tradeoff can choose to wait before retransmitting after a suitable number of failures. Alternatively, the link layer can give feedback to the routing layer that there is a latency-efficiency tradeoff, giving an opportunity to choose another link depending on the kind of traffic. Changing links introduces tradeoffs in fragment caching, as a receiver may not be able to distinguish a sender that is waiting due to a period of high loss or has chosen a new destination. Given the energy cost of communication and RAM limitations, these are difficult tradeoffs, and may benefit from packet control bits that indicate what policy the transmitter will follow.

Acknowledgment losses introduce an additional wrinkle in packet assembly. A node must have a mechanism for suppressing the resulting duplicates. If the sub-IP fragmentation and assembly layer does not have a cumulative acknowledgment scheme, then failed suppressions can lead to unnecessary packet delivery failures.

7 Conclusion

In this paper we have presented several key observations of low power 802.15.4 nodes. We have shown their implications to IPv6 routing over low power wireless networks. While we have not clearly illustrated what these algorithms and policies have to be, we have shown which of the policies currently used for other IP over wireless networks need modifications. The exact definition of these policies remains an open research topic. However, exploring the implications of low-power wireless to IPv6 routing is a first step to bringing IPv6 to PAN devices, which in the near future will be the most numerous class of networked nodes.

References

[1] Zigbee Alliance. `http://www.zigbee.org`.

[2] D. Aguayo, J. C. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *SIGCOMM*, pages 121–132, 2004.

[3] H. Balakrishnan, V. N. Padmanabhan, and R. H. Katz. The effects of asymmetry on TCP performance. In *Mobile Computing and Networking*, pages 77–89, 1997.

[4] Bluetooth SIG, Inc. `http://www.bluetooth.org`.

[5] A. Cerpa, N. Busek, and D. Estrin. Scale: A tool for simple connectivity assessment in lossy environments. Technical Report 0021, Sept. 2003.

[6] A. Cerpa, J. L. Wong, M. Potkonjak, and D. Estrin. Temporal properties of low power wireless links: modeling and implications on multi-hop routing. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 414–425, New York, NY, USA, 2005. ACM Press.

[7] D. Culler, P. Dutta, C. T. Ee, R. Fonseca, J. Hui, P. Levis, J. Polastre, S. Shenker, I. Stoica, G. Tolle, , and J. Zhao. Towards a sensor network architecture: Lowering the waistline. In *Tenth Workshop on Hot Topics in Operating Systems (HotOS X)*, 2005.

[8] D. Estrin et al. *Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers*. National Acedemy Press, Washington, DC, USA, 2001.

[9] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks, 2002.

[10] R. Govindan, E. Kohler, D. Estrin, F. Bian, K. Chintalapudi, O. Gnawali, S. Rangwala, R. Gummadi, and T. Stathopoulos. Tenet: An architecture for tiered embedded networks, November 10 2005.

[11] C. E. Koksal and H. Balakrishnan. Quality-aware routing in timevarying wireless networks. In *To appear in IEEE Journal on Selected Areas of Communication Special Issue on Multi-Hop Wireless Mesh Networks)*.

[12] N. Kushalnagar and G. Montenegro. 6lowpan: Overview, assumptions, problem statement and goals. IETF Internet draft, draft-ietf-6lowpan-problem-05.txt, August 2006.

[13] S. Lin, J. Zhang, G. Zhou, L. Gu, T. Hei, and J. A. Stankovic. ATPC: Adaptive transmission power control for wireless sensor networks. In *Proceedings of the Fourth International Conference on Embedded Networked Sensor Systems (SenSys"*, 2006.

[14] G. Montenegro and N. Kushalnagar. Transmission of ipv6 packets over ieee 802.15.4 networks. IETF Internet draft, draft-ietf-6lowpan-format-04.txt, August 2006.

[15] C. E. Perkins, E. M. Belding-Royer, and S. Das. Ad hoc on demand distance vector (AODV) routing. IETF Internet draft, draft-ietf-manet-aodv-09.txt, November 2001 (Work in Progress).

[16] D. Son, B. Krishnamachari, and J. Heidemann. Experimental analysis of concurrent packet transmissions in low-power wireless networks. Technical Report ISI-TR-2005-609, Nov. 2005.

[17] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. Understanding the causes of packet delivery success and failure in dense wireless sensor networks. Technical Report SING-06-00, 2006.

[18] The Institute of Electrical and Electronics Engineers, Inc. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), Oct. 2003.

[19] C. Won, J.-H. Youn, H. Ali, H. Sharif, and J. Deogun. Adaptive radio channel allocation for supporting coexistence of 802.15.4 and 802.11b. In *Proceedings of the 62nd IEEE Vehicular Technology Conference (VTC2005-Fall)*, September 2005.

[20] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 14–27. ACM Press, 2003.

Network System Challenges in Selective Sharing and Verification for Personal, Social, and Urban-Scale Sensing Applications

Andrew Parker* Sasank Reddy* Thomas Schmid* Kevin Chang* Ganeriwal Saurabh†
Mani Srivastava* Mark Hansen* Jeff Burke* Deborah Estrin* Mark Allman‡ Vern Paxson‡

*Center for Embedded Networked Sensing - UCLA †Google Inc. ‡ICIR/ICSI

ABSTRACT

We envision the blossoming of sensing applications in an urban context, enabled by increasingly affordable and portable sensing hardware, and ubiquitous wireless access to communication infrastructure.

In this paper, we describe the *Partisans* architecture, featuring infrastructure-supported selective data sharing and verification services. This effort represents an evolution of activity on embedded networked sensing from the scientific application space to applications in a space that raises novel issues in privacy, security, and interaction with the Internet.

1 INTRODUCTION

Application context inevitably drives the architecture design choices and the definition of services needed in a network. Over the past decade, the emergence of unanticipated applications of the Internet, such as peer-to-peer file sharing, networked gaming, podcasting, and voice telephony, has contributed to a pressing need to rethink the core Internet infrastructure and its accompanying architectural choices. To truly lay a foundation for tomorrow's infrastructure, however, requires going beyond simply reacting to applications that have already emerged, to proactively considering the architectural implications of new *classes* of applications. For example, embedded sensing will move beyond science, engineering and industrial applications to become an everyday tool for individuals and communities, enabling them to effectively observe distributed phenomena at personal, social and urban scales.

A key area in this regard involves embedded sensing technology, presently poised to move beyond scientific, engineering, and industrial domains into broader and more diverse *citizen-initiated sensing in personal, social and urban* settings.

By *sensing*, we mean, "the action of an *automatic device* in detecting, observing, or measuring something."[1] The sensing modalities we are considering include those available on cell phones and handhelds (imagery, video, audio), those typically used in environmental monitoring (temperature, pressure, light-level, etc.), and other modalities supported by mobile sensors.

Today, applications are emerging that draw on sensed information about people, objects, and physical spaces. Sensor-based applications enable new kinds of social exchange: by collecting, processing, sharing, and visualizing sensed information, these applications can offer us new and unexpected views of our communities and environment. To achieve their potential, these applications require fundamentally new algorithms and software mechanisms. The research described in this paper seeks to identify and develop an overall network fabric architecture that through various services coherently embodies such algorithms and mechanisms.

The applications considered in this paper can be divided into three categories that define how widely sensor data are shared: *Personal, social and urban*, which we sometimes refer to as *PSUS* (pronounced "pieces") applications. Medical monitoring is a good example of a *personal application*; observations about a patient's personal space, their heart rate or blood sugar levels, are only shared with the patient's health-care provider. By *social applications*, we mean situations in which data are shared among a group of participants, some, possibly all, of whom contribute data. Applications of this sort are best thought of as combinations of data services and "social networking software." Finally, urban-scale applications involve sharing data with the general public. For audio and imagery, we see precedents in podcasting and in photo sharing services like Flickr[2]. The scope of these applications is much larger and there might be an emphasis on identity control. In some cases participants may prefer to share data anonymously or "pseudonymously."

These emerging applications raise a host of important and challenging questions, whose answers potentially reach deeply into the network architecture.

- What mechanisms will enable those who deploy

[1] Oxford English Dictionary

[2] http://www.flickr.com

sensors to share data in a controlled way while respecting the privacy of those being sensed?

- Can we assure basic quality checks for data? For example, if a temperature reading is much higher or lower than readings taken from nearby locations, it should be flagged in some manner.

- By providing a suite of services, can we encourage responsible sensing practices?

We will argue that connecting these sensing systems to provide such basic assurances requires new infrastructure that we call *Partisans.* Such an infrastructure aims to provide the fundamental building blocks to aid in implementing a wealth of sensing applications, which in turn will promote citizen-initiated sensing projects.

2 DESIGN CHALLENGES

Data are more valuable if they can be *verified*. For example, a *subscriber* to a data feed might want to know, with some certainty, the time and *specific* location at which a measurement was taken. For some *providers*, such disclosure may be too invasive, and they would prefer to only reveal their location in terms of their ZIP code or county. The same kind of resolution control could apply to the time of a measurement, with some data providers choosing only to reveal the hour or day on which data were taken. Naturally, there will always be situations in which data can be shared freely, without restrictions. The emerging network of amateur weather stations is an example of this.[3] No matter what resolution a user is comfortable with, it is important that the context assigned to data be verifiable in some fashion.

By controlling the resolution or context of a measurement, the data contributor is, in effect, defining a privacy policy. We prefer to use the term "selective sharing" because it captures the idea that participants choose the conditions under which their data are divulged. For most of the examples presented so far, the sensing hardware acts in an essentially autonomous way, collecting data at regular intervals or in response to a detected event. Therefore, policies for selective sharing must be implementable as an automated component of a sensing system. Policies should also adapt in response to a contributor's changing public/private context.

Names touch on how we do dissemination, selective sharing, and verification. The items being named are the data streams published by sensor devices. For example, a mobile phone could have three data streams: an audio stream, a video stream, and a location stream (perhaps something fine grained like GPS, or coarse-grained like reachable cell towers).

[3] http://www.wunderground.org/

Personal, social, and urban sensing applications, as experienced and consumed by the end user, will straddle both traditional web-based applications, and sensor network applications. We envision a web services architecture that provides a platform to feed data to these applications, in much the same way that applications have sprung up around the Google Maps API and other platforms that give users access to vast amounts of data in a programmatic way over the web. RSS, ATOM and other web feed formats provide a useful, uniform interface to web sites that have stylized update mechanisms. This has enabled the construction of readers, aggregators, and other tools that enable the user to mix, filter, and otherwise experience content in customized ways. In a similar fashion, data streams from sensors should be subject to this kind of end-user manipulation.

2.1 Context Verification and Selective Sharing

Essential to building space-time semantics into the fabric is the ability for it to verifiably measure the location of a node and the time at which it transmits data. The basic measurement of time and location, while difficult under adverse conditions is conceptually simple and well studied. Using time-stamped message exchange based on protocols such as NTP, a node in the network can measure the clock offset relative to a sensor node [10] [5]. Likewise, a base-station in the network can measure distance to a sensor node using radio time-of-flight or signal strength for ranging, and then use multilateration to determine the position of the sensor node [12] using similar distance measurements made at other base-stations. Even simpler would be for the device to measure its own location and time using GPS. However, the crucial problem is one of verifiability: the location and time estimate must be robust to cheating by a malicious sensor node. We can guard against manipulation by an external adversary by having the sensor sign its data, but doing so requires a key distribution and validation infrastructure, which may run contrary to the large-scale and ad-hoc nature of the envisioned systems.

The physical context of sensor data is richer than just location and time. It includes, for example, the orientation of the sensor, measurements made by other sensing modalities, and measurements made by other sensors in the vicinity. Clearly, such additional physical context is of utility to the subscriber in interpreting the sensor data or in checking its integrity. For example, the utility of sound level from a directional microphone is significantly increased if the orientation of the microphone is also known. To increase the utility of sensor data, information about the sensor's context can be combined with statistical and physical models of how different sensing modalities are related, and of how measurements made by nearby sensors relate to one another. In-

deed, as the sensor infrastructure for PSUS applications proliferates, the increased spatial and temporal density of measurements will inherently provide additional physical context for the validation of a specific sensor measurement. Moreover, application deployments may have self-awareness sensors [7] whose purpose is to acquire information about the physical context as opposed to the phenomenon in which the subscriber might be directly interested.

The context derived from information provided by the sensors themselves is fundamentally different from location and time since the fabric has an independent ability and reason to measure space and time, but has no reason to directly measure the orientation of a sensor, the temperature in the vicinity of a sensor, etc. The role of the fabric in this case is therefore one of calculating and verifying the context according to application specified rules. For example, an application may request that the fabric corroborate a sensor reading with the readings from nearby sensors by comparing against their average. Though the application relies upon services provided by the fabric, the two are considered disjoint entities.

The final element in context verification is the ability for the application to exercise control over the context that is revealed to a subscriber. Specifically, the fabric will ensure that even if it has information about the physical context in fine detail, it does not send to a subscriber more contextual information than what the publisher is willing to share. The reason the fabric has access to higher accuracy data is so that it can better verify and aggregate it. For example, the fabric may know the location of a sensor to within a few meters, but the subscriber may only be willing to share the location information to the ZIP Code level. Likewise, a sensor may be willing to share information only as part of an aggregate in a geographical region. The fabric will deliberately reduce the fidelity of the context information it shares (location, time) or derives from sensor values. In addition, to combat emerging techniques for remote device fingerprinting based on measurements of timestamp drift [8] and localization using latency measurements [6], the fabric may add random jitter to packets.

An important question arises when a publisher can make available multiple versions of its sensor data that are blurred to differing degrees. We would like to ensure that the data remain equally valid regardless of resolution. For example, we would not want to allow a temperature reported as 28 C to also be published in a more blurred form as "in the range 20–25 C," as such inconsistencies can be used by the publisher to selectively skew the view of the data seen by 3rd parties (see below). Thus, for each given type of measurement we may require a process by which the fabric can (perhaps with the help of a neutral, external agent) objectively compare two versions of different precision.

Finally, leaving blurring up to the publisher imposes a potentially important limitation: it does not support forms of blurring that require blending together results from multiple publishers. For example, a publisher might be willing to contribute an observation only if at least N other publishers are contributing sufficiently similar observations, to resist fingerprinting of the publisher's identity based on the uniqueness of their data items. In this case, adequate blurring requires a group effort or a trusted intermediary. It remains to be seen whether we need this type of blurring often enough that we must revisit this facet of the fabric's architecture.

Besides physical context, also part of a sensor device's network context are network-level identifiers such as host name or IP address. To begin with, our approach would be to rely on the level of indirection provided by the fabric to optionally hide the sensor's network identity from subscribers.

2.2 Discovery and Publication

The naming and discovery service, which in many ways can be treated as a publish and subscribe mechanism, plays a similar role to that of the Domain Name System (DNS) in today's Internet. The service would map a tuple space of attributes to a handle (or set of handles) that may be used to operate on data streams. Attributes will typically consist of information such as the sensing modality, data format, location, orientation, etc.

There are two constraints that must be satisfied before returning handles to data streams. First, the attributes of the data stream and the attributes requested by the subscriber must match in some sense. There are some attributes that naturally form hierarchical relationships. Location is a primary example. These relationships must be known by the naming and discovery service. Second, disclosure rules accompanied by the data stream must be satisfied, which may take into account some aspect of the *subscribers* attributes (identity, location, and time).

Here again an important question of trust arises. Does the publisher trust the fabric to enforce the publisher's disclosure rules? Alternatively, we could propagate requests for streams all the way to the publisher to allow it to make the final decision. Doing so has the drawback that if the publisher (or its designated agent) is unavailable, then we must deny use of the data, even if it would otherwise be allowed. In addition, we would like to ensure that the publisher cannot bias the view of the data obtained by 3rd parties (such as competitors) by skewing which data elements the party sees, or their values. To combat this latter, we need the fabric to enforce the disclosure rules rather than the publisher; and, in addition, the rules themselves should be available for inspection to enable third parties to determine the fidelity and potential

biases of the data they receive.

Another challenge is to ease and encourage publication of sensor data by independent data providers, as well as application development by 3rd parties that pull on the published data streams. Primitive functionality should include aggregation, processing, and querying. The elements providing these services act as subscribers to data streams. In some cases, subscribers will act independently and crawl the network for available data streams, much like indexing services such as Google or Yahoo. In other cases, the sensor will task subscribers to aggregate data on its behalf, much like Flickr and blogging services.

3 System Architecture

3.1 Architectural Components

Our proposed architecture incorporates the entities listed below.

Sensors are data sources at the edges of the network. They may either be resource rich devices (e.g. [9]) directly resident on the network, or resource-constrained devices such as "Motes" [3], grouped many-to-one behind a resource rich proxy gateway node. We note that sensors can have roles beyond simply pure sources of data by providing control points to the external world for purposes such as configuring other sensors or otherwise acting on the physical environment.

Subscribers are sinks of sensor data. They may be either individual users interested in data streams and event notifications from sensors, or network applications that subscribe to sensor data and provide archiving, aggregation, distillation, signal search, and other such services to their clients. A physician may subscribe to medical events associated with a remotely monitored patient; or a smart home control software may subscribe to data from sensors deployed by the homeowner. Network applications acting as hosting services could allow users to share sensor data much like Flickr.com (images), Vimeo.com (video) and Odeo.com (audio). Once hosting services exist, it is sensible to posit a Google-like service that facilitates searching archived or "live" sensor signals.

Mediators are nodes in the network that provide (under application control) selected in-network functions on sensor data streams. These functions would include: enhancing streams with attested contextual information at a specified resolution; performing verifications on sensor data values such as range checks or comparisons with values at proximate sensors; performing anonymization of the streams by removing device identification information; replicating streams for delivery to different nodes; and providing reliability for intermittently connected sensor and/or subscribers. The functions themselves are performed based on disclosure and verification rules specified by the sensor. Moreover, the mediator would make use of trusted infrastructure for independently measuring the location and time of data sent by the sensor devices.

Distinct from efforts such as Active Networks, the mediators do not manipulate the sensor data values carried as stream payload, a choice motivated by the simplicity of not allowing complex applications to "program" the mediators. Transformation of sensor value streams for purposes such as anonymity preservation or for scaling to a presentation device is best delegated to the end points. (However, we still need external processes for determining that multiple versions of sensor values have equal validity, if not equal resolution, as discussed above.) We view the mediators to be like firewalls in terms of administration, deployment ubiquity, trust and transparency to the user, while being like distributed content caching servers in terms of inter-mediator coordination and hardware configuration. As a reseult, mediators will be geographically proximate to sensors that use their services. For example, a data provider using their university campus network for connectivity could assume (and through some investigation, verify) that they are "behind" a mediator administered by the school.

Registries are network entities that help subscribers discover and bind with sensor data streams. Their role is to provide a service analogous to that of the DNS, with a model of administration and deployment-ubiquity similar to DNS servers. Sensors register with the registry metadata information about the sensor data they publish, while subscribers use the registry to search for sensor data streams by querying over attributes such as location or type of sensor data. The registry maps the query via a tuple space search process to return a handle, or set of handles, for sensor data streams. We make the comparison to DNS as opposed to a search engine in order to emphasize the liveness and degree of control that sensors have over the data available in the registry.

3.2 Trust Model

The entity that has the most at stake is the data provider that is responsible for the sensor node. In its exchange with the mediator, the sensor may divulge higher resolution location, contextual, and sensor data for the purpose of aggregation and verification.. How best for the sensor to manage its relationship with the mediator remains an open question. The sensor also discloses potentially sensitive information to the registry, but the sensor must assume that once data is handed out to a subscriber, that the data is "out" and publicaly available. Thus, the registry disclosure rules should not be used for security purposes, but rather as a way to describe an appropriate matching against subscriber queries.

In general, the network, mediators, and registry are considered trustworthy by all, unless proven otherwise.

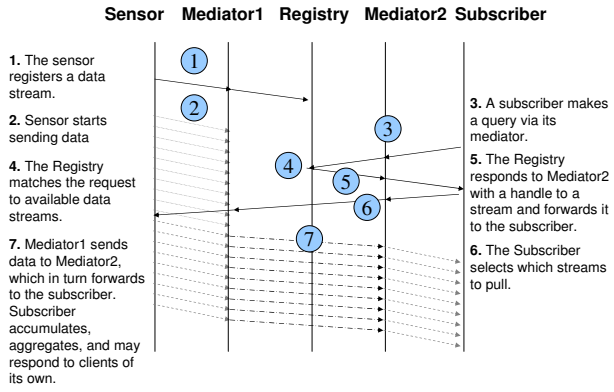


Figure 1: A sequence diagram of an example exchange between a sensor and subscriber.

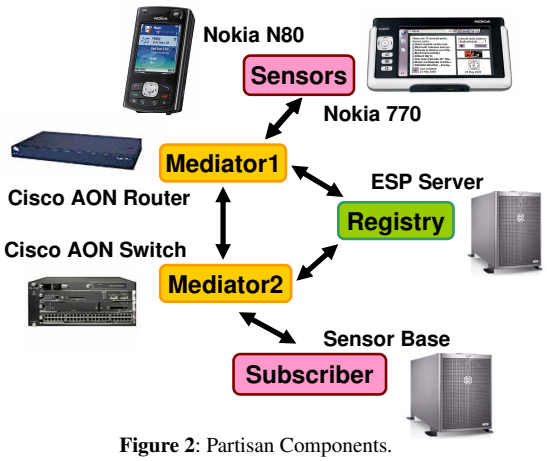


Figure 2: Partisan Components.

This is not significantly different from the trustworthiness afforded to similar components such as firewalls, DNS servers, email servers, etc. that are operated and secured by network administrators and service providers.

The trustworthiness of data received by the subscriber should be held with the same regard as any other data that a subscriber receives over the network. There are situations where an adversary has an incentive to advertise false data, and it is only through the availability of a sufficient number of honest components (sensors, devices, mediators, and registries) that a subscriber would hope to statistically verify the integrity of the data it receives.

3.3 An Example Exchange

Figure 1 depicts an example exchange between a sensor and subscriber. The steps below go into more detail.

Step 1. Upon deployment and configuration by its owner, a sensor registers the streams it is publishing with the registry service via Mediator1. The registration contains information about the sensor type, location, and context. It is these attributes that others will use to search and subscribe to sensor data streams. The registration will also contain disclosure and verification rules. For example, a sensor device could register (location= “Main Library”, type= “microphone”, format= “spectrogram”, UID= “3241531”). The sensor contributes to the location information in the data stream attribute, since the location may be at a finer granularity than to which the fabric is able to attest. If the sensor is mobile, it is the sensor’s responsibility to detect when it may have changed location and should bind with a new mediator and update the registry.

Step 2. The sensor now begins to publish its data to Mediator1, either proactively or when queried by that Mediator1 (as a consequence of a request by a subscriber).

Step 3. Sometime later, the Subscriber sends a query to the Registry through its mediator, Mediator2. The request contains disclosure rules as well. It needs to go through Mediator2 in order for the fabric to attest to the location (or similar attributes) of the requestor, as the sensor may have disclosure controls that are a function of requestor’s attributes.

Step 4. The Registry reconciles the disclosure rules of the sensor publishing the data with the request and attributes of the subscriber.

Step 5. The Registry returns a pointer to the matching sensor data streams to Mediator2. These pointers direct the holder to mediators that proxy the corresponding data streams.

Step 6. The Subscriber then directs Mediator2 as to which particular data streams to pull. (In the figure, there’s only one stream to pull.)

Step 7. Mediator2 then pulls the requested data streams from Mediator1.

3.4 Prototypes

We have prototype implementations of our four system components, as depicted in Figure 2.

The prototype sensor platforms are the Nokia N80 smart phone, and the Nokia 770 Internet Tablet. An example application is the EcoPDA (Ecological PDA) Project [2], an effort underway to assist in field observations for biodiversity and ecological research. With little modification, the EcoPDA project can satisfy the role of the Sensor in PSUS applications. The goal of EcoPDA is to automate or augment many aspects of mobile data entry (GPS location, voice recording, imaging) as well as prompt for and verify data input by the data provider. Additionally, the EcoPDA will upload to an aggregator, SensorBase [4] (described later in this section), on behalf of the data provider.

The prototype mediator rests upon the AON (Application Oriented Networking) platform. In [1], Sankar describes how AON platforms contribute to the architec-

ture of intelligent edge networks. The salient features of AON routers and switches are that they do application-level message classification, provide a declarative policy framework within which to operate on these messages (drop, cache, modify, forward), and offer an interface to easily inject policy into the network. We are currently developing software running on an AON Volant Blade to do simple location testimony/verification/obfuscation, as well as data aggregation.

The prototype registry is built upon the ESP framework. [11] ESP brings forward several concepts that are essential to managing, querying, and interacting with the wide variety of network sensing systems. The unifying interface language is ESPml. Sensor systems register themselves by describing their capabilities as an ESPml document. Agents can query the registry based on an area of interest and are returned an ESPml document that contains all the systems that match the query.

The prototype aggregator is SensorBase.org [4], which is a platform for common data storage and management for sensor networks. It provides users a web-service interface for publishing sensor network data. SensorBase also acts as a sensor network specific search engine, allowing users to query for specific data sets based on geographic location, sensor type, date/time range, and other relevant fields. Furthermore, the ability to search based on characteristics or features of the data themselves will soon be added.

4 Conclusion

We have presented an architecture for infrastructure supported selective data sharing and verification. Network testimony of when and where data is first injected allows mediating infrastructure nodes to execute selective data sharing on behalf of data contributers, and verification functions on behalf of data consumers. The result is an audit trail that plausibly verifies the integrity of the sensor data and some degree of context around that data (time and location). These services are a requirement for a rapidly emerging class of applications that draw upon sensed information about people, objects, and physical spaces.

There is a natural resistance to introducing yet more functionality into the network infrastructure. However, the services we propose (time and location testimony) are aspects of communication to which the network already has access; our proposal is to expose and utilize this information in novel ways. In principle it's possible to achieve some of the same effects of verification and selective sharing through end-to-end mechanisms (possibly with cryptographic techniques like zero knowledge proofs), but may not always be possible or practical. Thus, our architecture represents a trade-off between trust and practicality, taking into consideration the often-limited resources of the sensor.

References

[1] Cisco application-oriented networking blog. `http://blogs.cisco.com/AON/`.

[2] Ecopda. `http://www.lecs.cs.ucla.edu/urban-sensing/index.php/EcoPDA`.

[3] Tinyos: An operating system for networked sensors. *`http://tinyos.millennium.berkeley.edu`*.

[4] Chang, K., Yau, N., Hansen, M., and Estrin, D. Sensorbase.org–a centralized repository to slog sensor network data. In *DCOSS/EAWMS Proceedings* (June 17 2006).

[5] Ganeriwal, S., Kumar, R., and Srivastava, M. Timing Sync Protocol for Sensor Networks. In *Sensys* (Los Angeles, 2003).

[6] Huffman, S. M., and Reifer, M. H. Method for geolocating logical network address. In *United States Patent 6947978* (Sept. 2005).

[7] Kansal, A., and Srivastava, M. *Wireless Sensor Networks: A Systems Perspective, Eds. N. Bulusu and S. Jha*. Artech House, 2005, ch. Energy harvesting aware power management.

[8] Kohno, T., Broido, A., and Claffy, K. C. Remote physical device fingerprinting. *IEEE Trans. Dependable Secur. Comput. 2*, 2 (2005), 93–108.

[9] McIntire, D., Ho, K., Yip, B., Singh, A., Wu, W., and Kaiser, W. J. The low power energy aware processing (leap)embedded networked sensor system. In *IPSN 2006* (New York, NY, USA, 2006), ACM Press, pp. 449–457.

[10] Mills, D. L. Internet Time Synchronization: The Network Time Protocol. In *Global States and Time in Distributed Systems*, Z. Yang and T. A. Marsland, Eds. IEEE Computer Society Press, 1994.

[11] Reddy, S., Schmid, T., Parker, A., Porway, J., Chen, G., Joki, A., Burke, J., Hansen, M., Estrin, D., and Srivastava, M. Demo: Urbancens: Sensing with the urban context in mind. In *Ubicomp, to appear* (2006).

[12] Savvides, A., Girod, L., Srivastava, M., and Estrin, D. Localization in sensor networks. In *Wireless Sensor Networks*, C. S. Raghavendra, K. M. Sivalingam, and T. Znati, Eds. Kluwer, 2004.

Rethinking Wireless for the Developing World

Lakshminarayan Subramanian
Intel Research, Berkeley and
New York University

Sonesh Surana, Rabin Patra,
Sergiu Nedevschi, Melissa Ho, Eric Brewer
University of California, Berkeley

Anmol Sheth
University of Colorado, Boulder

ABSTRACT

Many rural regions in developing and developed countries with low user densities do not have good connectivity solutions. To date, networking research has largely focused on urban areas of the industrialized world. In this paper, we make the case for research on new appropriate wireless technologies that can provide low-cost, rapidly deployable connectivity solutions for low user-density regions. To this end, we compare and contrast the connectivity requirements that arise in the two domains and pinpoint the new research challenges that arise in low user-density environments. We describe our research efforts in this space and also share our initial experiences in deploying low-cost WiFi-based Long Distance (WiLD) networks in India, Ghana and the San Francisco Bay Area.

1 INTRODUCTION

Today, the evolution of networks in the developing world is taking quite an alternate route from the traditional networks we observe in the industrialized world. Many large cities in East Africa now have a large number of towers supporting a wide range of different long-range wireless technologies such as microwave, WiFi, WiMax and other commercial wireless broadband solutions. African countries see better opportunity in wireless options for regions that have low penetration of fiber and other wire-line connectivity solutions; many of these countries have higher cellphone penetration rates than fixed-line penetration [6]. The primary reasons for the boom in the use of wireless networks in developing countries are:

Lower cost: In developing countries, wire-line connectivity solutions are not economically viable in low-user density areas [7]. Satellite links, a common mode of Internet connectivity in much of Africa, are also very expensive and not widely affordable (typically US$2,000 per month for 1 Mbps). Establishing wireless distribution networks (microwave, WiMax, WiFi-based or CDMA450) to extend coverage within a region requires a much lower capital investment. This allows for decentralized rapid evolution of such networks by local entrepreneurs. Among different wireless options today, WiFi-based networks are *currently* much more economically viable than WiMax, CDMA450 and microwave.

Ease of deployment: Wireless networks are relatively easy and quick to deploy, particularly in cases where we do not need new towers. Networks in unlicensed spectrum are preferred because they can be set up by grass-roots organizations as needed, avoiding dependence on a telecom carrier. This is particularly important for rural areas, which are less enticing to carriers due to the low income generation potential,

Intranet usage: Providing network access does not necessarily have to be associated with Internet access. In many developing regions, basic local communications infrastructure is absent. A wireless network within a city or a district can enable a wide range of applications including telephony, essential services and health care. For example, we have deployed an intranet network in southern India between hospitals and rural vision centers that supports rural telemedicine [8].

Despite such a phenomenal growth in the adoption of wireless networks in developing regions, there have been very few research efforts that take a concerted view towards analyzing how to build such networks. The primary difference between urban environments in developed countries with a majority of regions in the developing world (with the exception of highly populated cities) is the *density of users*. We argue that prior work on wireless mesh networks [4] is best suited for urban environments with high user densities. At lower user densities, the type of wireless network best suited to provide coverage is significantly different from the mesh networking model; such a network would consist of nodes with directional antennas and point-to-point wireless links.

In this paper, we outline the research challenges that arise in building low-cost, long-range wireless networks for low density regions. Our research has primarily focused on WiFi-based networks given that WiFi is much cheaper than other wireless technologies and also operates in the unlicensed spectrum. Some of the early works by Bhagwat *et al.* [2] and Raman *et al.* [9] in this space focus on the specific aspects of tailoring the 802.11 MAC protocol to work in such settings; while this is indeed relevant, it represents a small portion of a much larger puzzle. In this paper, we take an end-to-end systems perspective at the overall challenge: how does one engineer a large-scale long-distance wireless network that can provide predictable coverage and good end-to-end performance in the face of competing traffic (from other sources using the same network) and over potentially highly lossy environments (induced by multi-path and external interference) and systemic link/node failures? Answering this question involves addressing challenges at various layers of the networking stack. In this paper, we elaborate on these challenges and describe our initial efforts towards address-

Characteristic	High User Density	Low User Density
Connectivity requirements	Full coverage required	Islands connected to each other
End Devices	Individual, mobile, low power budget and non-LOS	Shared, fixed, high power and LOS
Topology	Star-topology or mesh	Point-to-point with end points within the network
Applications	Mainly Internet access	Internet as well as peer-to-peer Intranet access

Table 1: Low user density and High user density region characteristics

ing some of these challenges. We also briefly describe our deployment experiences in building three such WiFi-based long distance networks in India, Ghana and the Bay Area.

2 Low vs High User Density Regions

In this section, we begin by contrasting low user density (rural and semi-urban) and high user density environments (urban) and make the case for point-to-point long distance wireless networks using directional antennas in low-density environments. We do so by pinpointing why other well-known wireless technologies (VSATs, cellular, mesh networks) are not economically viable in low-density environments. Next, given the distinction between these two environments, we describe the primary differences in the technical challenges that arise in point-to-point wireless networks in comparison to wireless mesh networks, which have received a lot of attention recently.

2.1 The Case for Point-to-Point Wireless

Figure 1 lists some of the fundamental differences between providing wireless connectivity in high user density and low user density environments. These differences mainly stem from the constraints of providing *low cost* wireless connectivity with small per-user cost and minimum or no recurring cost. In low density environments people are usually clustered around small localities (e.g. villages), with large distances among these clusters. Even within villages the user density is low compared to urban areas. In addition, the typically lower incomes lead users to share computer terminals (e.g. Internet kiosks) to amortize the relatively high cost of the devices and network connection.

Satellite networks provide fantastic coverage, but are very expensive. VSAT equipment installation costs over US$10,000 with a recurring monthly cost of over US$2,000 for a 1 Mbps link. In low user-density regions, VSAT is affordable only for businesses or wealthy users.

Networks with a base-station model such as WiMAX, and cellular networks like GPRS and CDMA, have an asymmetric design philosophy where expensive base stations are amortized by large number of cheap clients over many users. In low-density regions, such base stations simply do not cover enough users to be economically viable. The expectation that cellular solves the connectivity problem for developing regions is thus somewhat of a myth: cellular success in developing countries is an urban phenomenon, with a few exceptions. Bangladesh has good rural coverage because it is actually a very high density country, and base stations that cover roads and rail lines also cover many villages. China has dictated good coverage as policy, despite the economic issues. Other countries either subsidize rural users through taxation, much like the US universal access tax, or require some rural coverage as part of spectrum allocation. In its intended deployment model, with expensive basestations covering many users, WiMax also shares the shortcomings of other cellular technologies.

Finally, 802.11 mesh networks [4], also assume high user density. Moreover, mesh networks suffer from two basic problems when scaled to larger areas. First, as the network grows, an increase in the number of APs with omni-directional antennas leads to increased interference in overlapping cells. Second, the use of low-gain omni-directional antennas increases the hop length, and as a result throughput decreases. Bicket *et al.* [3] show that in Roofnet, longer routes (traversing multiple wireless hops) are disproportionately slower mainly due to inter-hop collisions.

Thus, we argue that for low density of users, approaches that provide full coverage are not feasible. The alternative would be to cover only those few places where connectivity is required, by employing long-distance point-to-point wireless links. Such links can rely on WiFi, point-to-point WiMax, or other technologies that support long-distance links offering reasonable bandwidths. In choosing such a technology, the most important factors are cost and configurability. An interesting case are environments that have a mix of low and high user density regions. Here, a combined approach where the mesh network is augmented by point-to-point links as required can also be considered ([5]).

Until now, for practical and cost-related reasons, we have chosen to examine the possibility of using WiFi-based Long Distance (WiLD) links. WiFi cards are cheap and highly available, enjoying economies of scale. In our existing WiLD deployments, the cost of a WiLD link is approximately $800 (excludes the cost of tower) with no recurring cost.[1] Because they operate in unlicensed spectrum, WiLD links are easy to deploy and experiment with, and spectrum license costs are eliminated. Manufacturers of WiFi chipsets (e.g. Atheros) often support open-source drivers, allowing us to completely subvert the stock 802.11 MAC protocol and tailor the protocol to meet our needs.

An alternative would be to use point-to-point WiMax links; such links would have a few important advantages over WiFi: configurable channel spectrum width (and consequently datarate), better modulation (especially for non-line of sight scenarios); operation in licensed spectrum would

[1] We are also deploying solar cells in our WiLD deployments

permit higher transmit power, and thus longer distances and better signal strengths. However, existing commercial WiMax products are only tailored for cellular providers and do not support point-to-point mode of operation. Existing WiMax hardware is more expensive than WiFi (about $10,000 for basestations), and the high spectrum license costs in most countries dissuade grassroot style deployments. Currently it is also very difficult to obtain licenses for experimental deployment and we are not aware of open-source drivers for WiMax basestations and clients (Wavesat offers a mini-PCI based WiMax client development kit [10]).

Consequently we advocate the use of WiLD links as the currently preferred solution; however, research investigating long-distance point-to-point wireless networking should be (for the most part) agnostic to the specific underlying wireless technology being used, allowing for other solutions to be used as they become available. We formulate our research challenges accordingly.

2.2 WiLD vs Mesh networks

We continue by discussing how the characteristics of WiLD networks differ from those of mesh networks, and thus lead to very different research agendas. We point out three key aspects that significantly differ between 802.11 deployments in low-density settings (WiLD networks) and high-density settings (mesh networks): external WiFi interference, multipath characteristics and routing protocol characteristics.

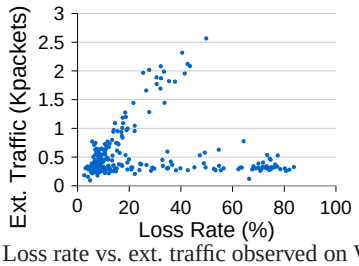


Figure 1: Loss rate vs. ext. traffic observed on WiLD link

External WiFi Interference: In settings where WiLD links co-exist with other external omni-directional WiFi transmitters (access points within the neighborhood), the hidden terminal problem is exacerbated. This is due to two features of WiLD links: directional transmissions and links with long propagation delays. Due to the highly directional nature of the transmission, a large fraction of interfering sources within range of the receiver act as hidden terminals since they cannot sense the directional transmission. However, in an omni-directional mesh network with overlapping transmission regions among neighbors, the fraction of external interfering sources that act as hidden terminals is much smaller. Due to long propagation delays, even external interfering sources within the range of a directional transmitter can interfere by detecting the medium as busy too late. Hence in WiLD settings, *any* external source can act as a hidden terminal.

Therefore, external WiFi interference can be a very important source of loss in WiLD environments; this is much less so in mesh networks. Figure 1 shows a scatter plot between the loss rate and the absolute number of external WiFi traffic frames received on an urban link over a period of 6 hours. The figure shows that a subset of the loss rate samples are strongly correlated with the external traffic.[2] This result is very different from the measurements reported in Roofnet [4] where the authors show the correlation between loss rate and external WiFi traffic to be very weak. Although these measurements are collected in urban links, they also directly apply in low-density networks where one of the endpoints is in an urban environment.

Multipath characteristics: In Roofnet [1], the authors conclude that multipath interference was a significant source of packet loss. However, in WiLD networks, we observe quite the opposite. This is primarily because the delay spreads in WiLD environments are an order of magnitude lower than that of mesh networks. The two factors contributing to lower delay spreads in WiLD networks are the long distance of WiLD links, and the line-of-sight (LOS) deployment of the nodes. The strong line-of-sight component in WiLD deployments ensures that the attenuation of the primary signal is only due to path loss, and most of the secondary paths are due to reflections from the ground. Furthermore, the long distance between the endpoints ensures that the primary and the secondary reflection travel almost the same distance, and hence reduces the delay spread. In comparison to our WiLD deployment, the Roofnet deployment has shorter links and non-LOS deployments, which significantly increases the delay spread.

Routing: From a topology perspective, two distinguishing factors between mesh and WiLD networks are that mesh networks are unplanned while WiLD networks are planned, and that the quality of links in mesh networks is time-varying and nodes have several neighbors to potentially forward packets. Hence, in mesh networks, routing is more opportunistic where nodes forward packets based on the quality of the link at a given time. Roofnet's routing protocol, Srcr, chooses routes with a minimum "estimated transmission time" (ETT) as a route selection metric [3]. In contrast, WiLD networks consist of a few dedicated point-to-point links and routing in WiLD networks resembles traditional routing protocols.

3 Existing Deployment

Currently, we have deployed several WiLD networks in India (a 9-link topology), Ghana (5 links) and the Bay Area in the US (7 links). We use these testbed deployments to understand the different research issues and to implement and evaluate the solutions to those challenges. The WiLD network in India connects several village-based vision centers

[2]Based on experiments performed in a wireless channel emulator we observed that at a channel separation of 2, the receiver is not able to receive the frames from the external interference source. However, the signal spillage of the interference source in the primary channel is sufficient to cause frame corruption. This explains why a subset of loss rate is not correlated with external WiFi traffic.

to the local Aravind Eye Hospital, and supports remote eye care as well as distance learning through interactive video conferencing. In Ghana, the links are used by the University of Ghana to share Internet access, for distance learning, and to exchange electronic library information among its different campuses. Distances of our WiLD links vary from 10–80km with relays installed where there is not line of sight due to geographical limitations.

We use low power single board computers (SBC) with a 266 MHz x86-based chip, 128 MB RAM and up to 3 wireless cards for our wireless routers. For radios, we use off-the-shelf high power 802.11a/b/g Atheros cards with up to 400 mW of transmit power output. The platform runs a stripped down version of Linux from a 256 MB CompactFlash card. To form long distance links we use high gain parabolic directional antennas (24 dBi, 8 degree beam-width). In multihop settings, nodes can use multiple radios with one radio per fixed point-to-point link to each neighbor.

The above choice of hardware enables us to design low-cost routers (less than $400) that consume less power (5–10W) and are of low weight (10–15 kg per node with two antennas). While the small size and weight allows us to use less expensive guyed-wired towers, the low power consumption means that we can use small solar panels, which reduce the operating cost and increase reliability when uninterrupted grid power supply is not available.

4 Research Challenges

In this section, we elaborate on the research challenges that arise in engineering large-scale WiLD networks to achieve predictable end-to-end performance in the face of competing traffic from other sources and highly lossy links (induced by external interference). We classify the research challenges into the following categories: (1) MAC layer challenges; (2) Loss recovery mechanisms; (3) QoS Provisioning; (4) Troubleshooting, reconfigurability and management; (5) Network planning and deployment. Associated with each of these challenges, we describe some of our early efforts to address them.

4.1 MAC Layer Challenges

The first challenge in running 802.11 on long-distance multihop links is to adapt the 802.11 MAC protocol [9] to overcome its fundamental limitations which can be summarized as:

• ACK timeouts: The simple stop-and-wait recovery mechanism of the stock 802.11 protocol requires each packet to be independently acknowledged. This recovery mechanism is ill-suited for long propagation delays, as it limits utilization and thus bandwidth. Worse, if the time taken for the ACK to return exceeds a card-specific maximum timeout, the sender will retransmit unnecessarily and waste bandwidth .

• Collisions due to bidirectional traffic: The CSMA/CA channel-access mechanism is not suitable for long distance links; listening at the transmitter reveals little about the state of the receiver, due to the long distance and stale carrier sense information due to propagation delays.

• Multi-link Interference: When multiple WiLD links originating from a single node operate on the same or overlapping channels, the transmission of one link can interfere with packet reception on other links, because local side lobes are of similar strength to the signal received from afar.

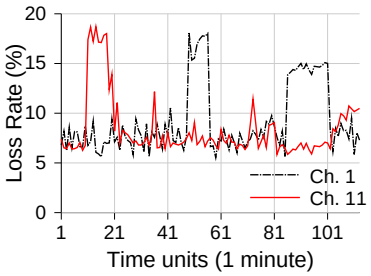


Figure 2: Loss variation over time across channels 1 and 11

TDMA MAC Protocol with sliding window: The above limitations of the stock 802.11 MAC protocol motivate the need for a TDMA-based MAC protocol that synchronizes the transmissions from the endpoints of a single point-to-point link. For a node having multiple outgoing point-to-point links, Raman et al. [9] propose having *simultaneous send* and *simultaneous receive* to eliminate interference. In addition, the stop-and-wait recovery mechanism of 802.11 is unsuitable. We implement a sliding-window based flow-control approach with the TDMA slots.

TDMA Slot Scheduling: Given these constraint of simultaneous transmit and receive, finding a feasible TDMA slot schedule in a multihop network is non-trivial especially if we want to achieve optimal throughput across the whole network. However, it can be shown that for bipartite graphs, we can always find such a slot schedule.

4.2 Loss Recovery Mechanisms

Across all of our WiLD networks, the presence of external WiFi interference results in very high loss rates on WiLD links. Furthermore, due to the long distances, the extent of interference could be very different at the two ends, making WiLD links asymmetric. Also, it is common to have links with loss rates fluctuating between $5 - 80\%$ over short time scales.

Figure 2 shows the loss rate sampled every 1 minute across channel 1 and 11 for a 20 km WiLD link. The figure shows that both channel 1 and 11 have long bursts of high loss rate due to external interference. Even in absence of long bursts, a residual 5–8% loss still exists. Given the situation, an important challenge is to device appropriate link level loss recovery mechanisms that can achieve predictable performance in the face of high loss variations.

Retransmissions with Bulk ACKs: The first approach for loss recovery is where the receiver acknowledges a set of frames at once using bulk ACKs, in the sliding window set-

ting proposed previously. The lost packets are then retransmitted accordingly.

Figure 3 shows the comparison of bidirectional TCP throughput achieved at various distances by the stock 802.11 MAC protocol (using CSMA) and by our implementation of the TDMA MAC protocol with bulk ACKs. To emulate long distances, we use a wireless channel emulator. We can see that as the distance increases, the throughput of CSMA MAC decreases gradually until 110 km, which corresponds with the maximum ACK timeout, and then it drops drastically. However, the TDMA MAC protocol using bulk ACKs provides sustained high throughput even at very long ranges.

Adaptive FEC: With such highly variable packet losses such as shown in figure 2, the retransmissions based approach would give us 0% loss but with highly variable delay and this is not suitable for audio and video traffic. We therefore propose an adaptive FEC based loss recovery mechanism which limits the delay experienced at each hop while guaranteeing a small loss rate. We are currently investigating appropriate FEC coding mechanisms for our WiLD setting. We observe that the loss variability of the WiLD links are very hard to predict, making the problem of determining the appropriate FEC recovery mechanism a challenging one.

4.3 Quality of Service

Many applications that use WiLD networks require QoS (*e.g.*, video-conferencing sessions in rural telemedicine). Unlike the case of the Internet architecture, in WiLD networks we have the flexibility of modifying routers to implement QoS mechanisms. However, many of the traditional QoS mechanisms do not blindly carry over due to peculiar constraints imposed by WiLD networks. First, unlike traditional wired links, WiLD links cannot be characterized by a fixed bandwidth value. In the presence of high loss variations, the available bandwidth (after recovery) is time-varying. Also, the need for synchronous packet transmissions and receptions at a node, creates a direct coupling between the available bandwidth on adjacent links; in other words, any variation in the slot size along one link, affects the one-way bandwidth on adjacent links. Second, WiLD networks experience highly-variable delays due to the TDMA nature of packet transmissions coupled with loss recovery. Hence, providing end-to-end bandwidth and delay guarantees for flows requires scheduling mechanisms that can take into account the variable link bandwidths and link delays. Traditional QoS mechanisms assume the concept of *flow isolation* i.e. once a set of resources are allocated to a flow, this flow is unaffected by competing flows. This assumption does not completely hold in WiLD settings since the introduction of a new flow can potentially affect the resource allocation of competing flows (either along on links in the same path or adjacent links along the path).

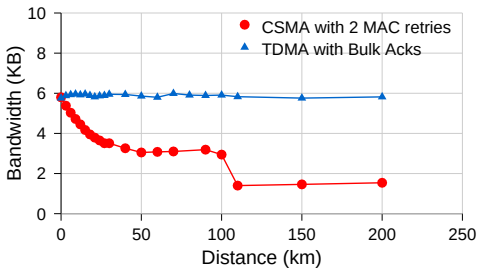


Figure 3: Comparison of WiLD MAC and stock 802.11 MAC

In addition to these differences, WiLD nodes have constraints on processing power (266 MHz) and memory (128 MB) that may rule out many fancy strict/statistical QoS mechanisms which would require nodes to maintain per-flow state and track per-flow usage. We are currently deploying simple QoS mechanisms based on *traffic priority classes* similar to Diffserv without supporting any form of strict guarantees. To provide statistical guarantees at a per-hop level, the primary link-layer parameters that we can manipulate are: (a) loss-recovery parameters (FEC, retransmissions); (b)varying the TDMA slot-size to reduce delay. Manipulating these parameters represents a trade-off spectrum between achieved loss-rate, delay characteristics, available bandwidth. As part of future work, we plan to analyze this trade-off spectrum and quantify the achievable QoS properties in WiLD environments. Another related problem is the *optimal TDMA scheduling problem*: Given a *traffic demand matrix* between various sender-receiver pairs, can we compute an *slot schedule* for every link in the network that can satisfy all the traffic demands?[3]

4.4 Troubleshooting, Reconfigurability and Management

A key aim in WiLD networks is to reduce the operational cost of maintaining the network. This is critical due to the lack of trained manpower in many developing countries, and long delays involved in accessing the endpoints which are usually tower mounted and could be separated by large distances.

Our experience with WiLD deployments shows that the network can malfunction in a number of ways ranging from complete failure of links (hardware board failure, corruption of the flash memory cards, lightning strikes), to performance degradation over time (from misalignment of antennas, signal attenuation from rain water clogging RF cables, interference from external sources).

Reconfigurability: One way to deal with complete failure of links or nodes is to design a redundant network topology, with more than one possible path between the wireless nodes. To reduce the cost of additional redundant links we are exploring the use of low-cost *electronically steerable antennas* instead. On a link failure, these antennas can dynamically realign themselves and reform the topology of the network to route around failed nodes or links such that network connectivity is maintained.

[3]This problem assumes that all links are in the same channel. Given non-overlapping channels, one can imagine a similar problem coupled with the need for an appropriate channel allocation mechanism.

Safe Upgrades: A *safe upgrade mechanism* is also required for changing either the firmware or even the network configurations on the routers. Any failure during this process could lead to the endpoints being disconnected and out of reach. To avoid such failures, we use the built-in hardware watchdog timer to power cycle the router on a failed kernel change or erroneous configuration change and revert to a default "golden" version.

Monitoring: The challenge in network management is to continuously monitor the network with both passive and active measurements to test for anomalous behavior. Additionally, the data aggregated from the distributed end-points in the network should be automatically analyzed to pin-point the location of the fault as well as diagnose the root cause of the fault. This information should be provided to the semi-skilled network administrator in a human readable form with concrete troubleshooting steps to perform.

Currently, in our existing deployments, we periodically initiate reverse ssh tunnels from the wireless routers to our server in Berkeley to collect a high level periodic health summary of each router node in the network. An alternate solution is to have a completely *orthogonal communication channel* like GSM/SMS. They provide a backup path for rare situations where a remote reboot is required, but are expensive and assume some form of cellular coverage.

4.5 Planning and Deployment

Planning of WiLD networks needs much more careful consideration compared to mesh networks with omnidirectional antennas. Since WiLD links traverse long distances, they require line of sight for operation; this usually implies towers at each end. As the towers compose a substantial part of the total cost of the network, the challenge is to select the locations of sites and the links so that the overall cost of the towers is minimized (determined by the heights of the towers). Site selection is also influenced by the presence of external WiFi interference, as well as interference from the nodes which are part of the WiLD network. WiFi interference from the nodes within the network as well as from the external sources can be minimized be judiciously selecting the transmit power of the nodes. By over-provisioning the signal at the receiver, capture effect can be used to eliminate most WiFi interference.

An additional significant problem in the deployment of WiLD networks is the difficulty of performing accurate manual alignments of the directional antennas for each long distance link. This is exacerbated by the fact that factors like wind and wear and tear of towers can cause the antennas to further misalign over time. In this respect, electronically steerable antennas can be used for automatic alignment. The open research challenge lies in devising efficient algorithms to discover peer nodes and maintain alignment using continuous adaptation over time.

5 Non-technical challenges

While deploying wireless networks in developing countries we encountered a variety of non-technical problems. These deployments present much larger installation, maintenance and servicing costs, due to lack of local technical expertise, equipment availability and logistics. Consequently, there is a need for production-quality solutions, and not just research prototypes. The hardware and software must be robust, user friendly, and simple to install, maintain and manage. Local partners must be trained as well. Our group has learned these lessons the hard way in India and Ghana.

Another barrier is local telecommunication regulation, which is hindered by limited technical staff, "imperfect" government, and the presence of local incumbent monopolies. Some of the problems we encountered are: restrictions on using VoIP (favoring local telecom monopolies), licensed or even restricted frequency bands that are unlicensed everywhere else in the world, and unregulated wireless usage resulting in significant same-band interference.

6 Conclusion

We argue the need for concerted research efforts to develop cost-efficient networking solutions for providing connectivity to regions with low user densities. To this end, we examined various wireless options and their suitability, and explored WiLD networks as a promising option. By taking a broad view of the problem, we found challenges at essentially every layer of the network and thus a range of areas for new research.

References

[1] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level Measurements from an 802.11b Mesh Network. In *ACM SIGCOMM*, 2004.

[2] P. Bhagwat, B. Raman, and D. Sanghi. Turning 802.11 Inside-out. *Hotnets-II*, 2004.

[3] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and Evaluation of an Unplanned 802.11b Mesh Network. In *ACM MOBICOM*, 2005.

[4] S. Biswas and R. Morris. Opportunistic Routing in Multi-Hop Wireless Networks. *Hotnets-II*, 2003.

[5] Dharamsala Wireless-Mesh Community Network. http://www.tibtec.org.

[6] International Telecommunications Union. World Telecommunications/ICT Development Report. 2006. http://www.itu.int/ITU-D/ict/publications/wtdr_06/.

[7] S. M. Mishra, J. Hwang, D. Filippini, T. Du, R. Moazzami, and L. Subramanian. Economic Analysis of Networking Technologies for Rural Developing Regions. *Workshop on Internet and Network Economics*, 2005.

[8] R. Patra, S. Nedevschi, S. Surana, A. Bakre, E. Brewer, K. Fall, and L. Subramanian. Achieving High Throughput in Long-distance Wireless Networks. *Intel Technical Report*, 2006.

[9] B. Raman and K. Chebrolu. Design and Evaluation of a new MAC Protocol for Long-Distance 802.11 Mesh Networks. In *ACM MOBICOM*, 2005.

[10] Wavesat. WiMax 3.5GHz mini-PCI Reference Design Kit. http://www.wavesat.com/products/mini-pci.html.

Service Portability

Why http redirect is the model for the future

Sumeet Singh Scott Shenker George Varghese

Abstract

The Internet provides tremendous flexibility, in that it can support a wide variety of services, and accessibility, in that these services can be invoked from anywhere. However, the current Internet architecture does not easily support service portability. *If users want their service names to be persistent then they must stick with the same service provider because service names, such as email addresses, are tied to administrative domains.*

In this paper we present a system called Permafind that gives users a persistent name for their services while allowing them to switch among service providers. Permafind applies to a wide range of services, and is immediately deployable. Serendipitously, Permafind also allows dynamic service insertion thus permitting many of the capabilities of more revolutionary proposals such as i3. Permafind embodies no technical innovation, but it does suggest that the notion of redirection, as embodied in HTTP, is a crucial feature for future service protocols.

1 Introduction

The Internet, through the generality of its architecture and the ubiquity of its deployment, provides a flexible, and pervasive communication platform. This fertile electronic soil has given rise to today's thriving ecosystem of Internet services. This evolving ecosystem has gone through at least three distinct developmental phases. In the beginning of the web, most Internet content was provided by individuals and nonprofit organizations. It took a few years for the commercial entities, such as news organizations and banks, to believe in the Internet but once their doubts were allayed, they adopted the web with a vengeance. Their adoption ushered in the second, more commercial, phase of the Internet, which has seen a new generation of corporate giants such as Yahoo!, Amazon, and Ebay, arise *de novo* from the Internet froth.

In recent years it appears that we are entering a third phase, sometimes called Web 2.0, in which an ever-increasing variety of *personal* services and content are being offered, though often hosted on commercial platforms. For instance, Myspace and YouTube have become extremely popular in recent years, and these are in addition to instant messaging, blogging, electronic mail, and web pages. While it isn't clear which category of content, commercial or personal, dominates the Internet, the hallmark of the last few years is the dramatic emergence of the latter.

The rise of these personal services presents a problem, one that already existed with ancient services such as email and web pages, but is now exacerbated by the proliferation of other modes of expression. While some adventurous individuals host their own services, the vast majority of these personal services are hosted by third parties, such as commercial providers, employers, or other organizations whose resources are not under control of the individual. There are a variety of reasons why users would like to move their services to a different hosting platform — such as pricing, change of employment, better service, etc. — but the current environment makes this difficult. Put concretely, users would like to *easily* switch their email from yahoo.com to gmail.com, and their blogs from blogger.com to newblog.com, as needed, without manually contacting all possible email correspondents or blog readers.

There are two main barriers. First, the Internet naming system ties services to administrative domains. Mail sent to an email address of sally@company.com will be delivered to a mail server controlled by company.com, and if Sally moves to a new company there is no way for this email address to follow her there. Second, despite years of effort there is no effective Internet directory that can serve as the "phone book" of the Internet. Search services do an amazing job with web pages, but for other services they perform significantly less well.

Thus, when someone moves their blog from Apcala to zoomshare, there is no way for a dedicated reader to know unless the person leaves a forwarding message. This also applies to email, which must be forwarded, and web pages, which must be redirected. In each case, the person must rely on the kindness of, or payments to, the previous hosting platform to maintain their forwarding address. This has created a significant barrier to service *portability*. Users tend to keep their services where they are, even when better opportunities appear, because of the significant inconvenience of moving to a new platform.

Given the role the Internet has played in disintermediation, which greatly increased the transparency and directness of many services, it is both archaic and ironic that for some of the Internet's most basic services, such

as email and web, individual users face such significant barriers to movement.

There are myriad papers that describe changes to the Internet architecture that would solve this problem (see, for example, [1–3, 5–7, 9–11, 13]). For instance, portability could be provided by a level of indirection, so that the name of a person's service is persistent but always resolves to the current provider. Similarly, an effective directory service would greatly alleviate the problem. However, neither of these are likely to happen any time soon, so our goal in this paper is to describe a solution that, while neither ideal nor elegant, is deployable in the short term. Beyond this short term focus, we are also looking for an approach that will push the architecture in a better direction in the long-term. To that end, we explore the basic principles one would need to follow when designing a protocol that would allow a service to be more portable. It turns out that HTTP, with its ability to redirect, is a model for such protocols [4].

We call the resulting system *Permafind*, and have a prototype available for experimental use (we encourage the reader to visit *www.permafind.com* and use the invite code 222 to register; this system is similar to the IKI web site http://http://www.iki.fi/index.html). While there is no ingenuity in the component mechanisms (indirection, redirection, relaying, and proxying), our intent is to turn a combination of these component mechanisms into a more universal mechanism that can be seamlessly invoked for all applications.

A side benefit of this approach is that it naturally enables a limited form of *service insertion*. For instance, one can have one's email sent through a spam filtering service of one's own choice.[1] The ease with which users can access third-party services might give rise to a much richer set of such services, thereby enhancing user functionality.

2 Designing for Portability

In this paper we set ourselves the following goals for our solution to service naming:

- *Persistence*: Each user should have a persistent, and therefore provider-independent, name for his/her personal services, ensuring that others can always reach them.
- *Generality*: To avoid separate *ad hoc* approaches for every application, the solution should work with traditional applications, newer Web 2.0 applications, and also future applications. In particular, future applications may run on top of other protocols than TCP or HTTP, so that one cannot rely solely on techniques embedded into these protocols (such as HTTP redirect).
- *Incremental Deployability*: The solution should be deployable without changing existing structures. In particular, services using this solution should be accessible by users of existing browsers and unmodified hosts.
- *Performance*: The portability solution should not cause significant loss in efficiency or increase in cost, both of which might deter its use by providers and users. Relevant performance measures include user-perceived latency and server throughput.
- *Ease of use*: The service should be usable by naive users, so it can't require users to run their own servers or configure their own DNS records.

These goals impose several constraints. Clean-slate designs, such as those based on flat and self-certifying names (*e.g.*, [13]), are ruled out by the need to be incrementally deployable. Thus we must provide persistent service identifiers based on the current naming system (domain names resolved by DNS). This would suggest an approach where each user has a personal and persistent domain name; *e.g.*, Bob could have a domain name bob.org (or, more likely, bob1753.org). To achieve generality, it must apply across all applications; for this we could use a hierarchical naming scheme such as:.

web pages: http://www.person.domain/path
blog: blog.person.domain
email: name@person.domain

The use of naming conventions such as these does not, by itself, solve the incremental deployability problem. Take the case of Bob Smith's blog, blog.bobsmith.org. If we look it up in DNS, we can get an IP address, but the requesting client (the one trying to reach Bob's blog) doesn't want an IP address; it needs the result to be translated (in the case of blogging) to a URL such as blogger.com/bobthebuilder. The problem is that the name given to Bob at blogger.com (*e.g.*, bobthebuilder) may be specific to the provider blogger.com and is thus not invariant. Unfortunately, DNS as it exists today only translates domain names to IP addresses, not URLs. Application-specific hacks like MX records only translate application-specific requests to an application-specific IP address (*e.g.*, returning the IP address of a mail server rather than the IP address used for a web page at that domain name). Such DNS modifications do not allow translation to additional provider-specific service tags such as usernames.

From a technical standpoint one could easily modify DNS so that it could return this additional information. Unfortunately, this approach not only requires substantial changes to DNS, it would also require all

[1] This feature is already available, such as in acm.org email, but our emphasis here is that Permafind explicitly sets itself up as a broker for these third-party services, allowing user-specified service insertion on a per user-name basis, as opposed to offering a single inserted service as a side-benefit.

clients (such as browsers) be modified in order to understand this new information. For instance, an unmodified browser trying to connect to Bob's blog will try to resolve bob.blogger.bobsmith.org and expect an IP address in return. Even if DNS resolves bob.blogger.bobsmith.org into blogger.com/bobthebuilder, the existing browser will treat this returning data as an IP address and fail.

Given that we have to stick with current DNS semantics, the next step in our search for a solution is to continue to use service-specific DNS names (as in bob.blogger.bobsmith.org), but instead of resolving them at the DNS level (*e.g.*, providing IP addresses), we generate application-level responses. To illustrate, consider the domain permafind.com which we have adopted for use in our system. A user that wishes to deploy a portable service (*e.g.*, email, blog) is given a persistent name bobsmith.permafind.com. Assuming Bob wishes to deploy a blog and email, he does so using two separate names email@bobsmith.permafind.com and blog.bobsmith.permafind.com.[2]

The trick is to have the Permafind server act as both a regular DNS server (to translate standard service requests to IP addresses) as well as an application server. The Permafind DNS server resolves DNS requests to itself. When the application-level request is then sent to the Permafind server, it attempts to direct the request to the appropriate server name. To do this, Permafind builds on three basic primitives: *redirection*, *relaying*, and *proxying*. In redirection, the Permafind server responds to an application request with a redirect message containing the appropriate service invocation. The key here is that this occurs at the application-level, which understands the appropriate semantics (such as the inclusion of a username in a URL). For applications that don't understand redirection, the Permafind server relays the request to the appropriate location. For applications that support proxying, the user can choose to have the Permafind server act as proxy; this will, as we will discuss later, allow the user to invoke further processing on the returning data. The Permafind server maintains a table, based on user input, of the appropriate mappings between the persistent Permafind service names and the current service locations.

We illustrate this with three concrete examples:

Redirection: Joan accesses Bob's blog by typing in the URL blog.bobsmith.permafind.com into her browser. The browser uses DNS to resolve blog.bobsmith.permafind.com into the IP address of the Permafind server. Joan's browser then sends an HTTP request, to which the Permafind server responds with an HTTP redirect pointing to the current URL, blogger.com/bobthebuilder.

Proxying: This is like relaying except that the Permafind server, rather than returning an HTTP redirect, forwards the request on as a proxy and receives the returning data. Proxying allows a user to specify, in addition to the location of his current service, any additional third-party functions that should be applied to the data. For example, for Instant Messenger services this could involve translating the IM request from one format (say Yahoo) to another (say AOL).

Relaying: Joan sends email to Bob at email@bobsmith.permafind.com. DNS resolves the MX record to the Permafind server. When the email arrives at this address the Permafind mail server relays the mail to the appropriate mail address, say bob@gmail.com.

Almost all current service interfaces support one of these three primitives. Moreover, the series of indirections (whether redirection, proxying, or relaying) may not be expensive in terms of throughput or latency. Thus, this approach is general, incrementally deployable, and reasonably efficient. It is also extensible; if a new breed of application X uses its own protocol rather than HTTP, Permafind will only have to add an application X server to do application X level redirection, proxying, or relaying. Since Permafind need only support a small subset of application X (enough to invoke relaying, proxying, or redirection), this is not very burdensome. We now discuss these mechanisms in more detail.

3 Detailed Design

The Permafind service allows users to sign up for subdomains of the permafind.com domain name, over which they have autonomous control. For example bob may register the sub-domain bobsmith, thus giving bob control over how the FQDN bobsmith.permafind.com is resolved at the permafind.com DNS server.

A Permafind name is processed in two steps.

DNS resolution: Clients use DNS to resolve the FQDN into the IP address of the permafind.com server. A easy and efficient solution to achieve this is to create a wildcard DNS entry in the DNS zone file for the permafind.com server. Using the wildcard DNS entry (allowed by all popular DNS servers), all sub-domains of permafind.com are automatically resolved to the IP address of permafind.com. This ensures that there is no delay for name resolution, so once a user reserves a particular sub-domain on the permafind.com server it is visible instantaneously.

Application-level translation: After the client receives the IP address from the DNS resolution, it then submits its application-level request to the permafind.com server using the user's service-specific FQDN (such as email@bobsmith.permafind.com and

[2] We use the formulation bobsmith.permafind.com as an example; it is easy to extend this to allow bobsmith to have several "accounts" attached to the bobsmith name, for example bobsdog.bobsmith.permafind.com

blog.bobsmith.permafind.com) . The corresponding service running on the Permafind server maps the service request to one appropriate for the current service location using information previously provided by the user. To remain compatible with existing clients, Permafind only maps an input name to an output name of the same type (*e.g.*, from URI to URI, or from email address to email address). The mapping also indicates whether a redirect or relay service is used.

If service insertion is used, the mapping provides the next stage in the chain of inserted services. Once again, redirect or relaying could be used to direct the data to the first stage in the service chain. For email, there is an elegant solution (that requires no changes at intermediate service providers) using a different Permafind name (per user) for each provider in the service chain. Figure 1 shows an example of service insertion for a receiver R (with Permafind address $R@P$) who requires the insertion of a spam filter (with Permafind address $R_1@P$ for user R) and a Virus checker (with Permafind address $R_2@P$ for user R)). $R_1@P$ and $R_2@P$ are internally assigned by Permafind to allow the same mapping database to facilitate both portability and service insertion. Unlike the insertion of Postini spam filtering by acm.org, the design allows *user-specified* service insertion on a granular *per-user* basis.[3]

For HTTP based services, Permafind will need arrangements with each service provider in the chain to relay to the next service. Difficulties with end-to-end semantics caused by mechanisms like cookies make it difficult to do general service insertion for HTTP based services. However, service insertion based on the initial control messages (*e.g.*, URL filtering) is easily possible.

In the actual implementation, the Permafind webserver maintains a mapping table using a MySql server for fast and uniform access. The current prototype has only a URI server and an email server. For all incoming HTTP requests, a local SQL query is made using the FQDN supplied in the URI and the resulting mapping is sent to the client using the HTTP-302 Found directive. HTTP-301 (Permanently Moved) will not work because clients would then bypass Permafind in the future. HTTP redirection ensures reachability for all popular Web 2.0 services such as blog, rss, podcasts etc. as well as traditional web-pages and thus allows incremental deployment.

The Permafind email server uses relaying instead of redirection. While redirect is part of SMTP, not all email servers support redirection correctly. Relaying also hides the final destination email address from the sender. Commercial providers such as acm.org and gmail.com already support email relaying so we will not elaborate further.

[3]Service insertion is not implemented in the current prototype.

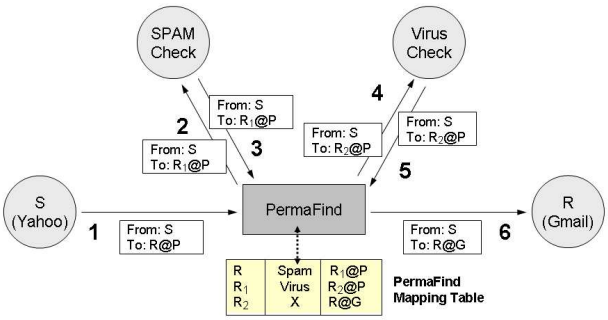


Figure 1: Steps in processing an email message from S to receiver $R@G$ with spam filtering and virus checking as inserted services. $..@P$ denotes a Permafind address. In Step 3 the message comes back from Spam Checking with $R_1@P$ as the To address. Permafind uses the mapping table (indexed by $R_1@P$) to direct the message to the Virus Checker with a To address of $R_2@P$. This in turn causes the virus-checked message to be forwarded to $R@G$ via Permafind.

Besides redirection and relaying, we anticipate a Permafind server being used as an *application level proxy*: a gateway that intercepts and rewrites control and data messages, thus providing additional services. For example, one gateway could convert between Instant Messenger protocols. A second gateway could translate between entirely different protocols and modalities, for example converting voice messages to email messages.

4 Looking Towards the Future

Permafind uses a set of standard mechanisms (relaying, redirection, and proxying) and one level of indirection (a very old idea in computer science). There are already commercial email services, such as acm.org, which offer relaying as a service and the IKI site offers both relaying and redirection. So what was our point in writing this paper?

In our defense, the current situation is far from ideal. These techniques are currently configured, deployed, and invoked on a per-application basis. For example, acm.org does not provide a web redirection service for say blogging or photo swapping. By contrast, Permafind places these methods in a unified framework, requiring no configuration on the client and straightforward account management (to keep mappings up-to-date) by the user.

Beyond this technical unification, the *combination* of indirection with redirection at the indirection point appears to be more powerful than indirection or redirection in isolation. Clearly, the intent of redirection was to allow portability, but redirection at the old service location is more problematic than redirection at an indirection point such as Permafind. Similarly, indirection followed by relaying is less efficient (because all the data passes through the relay) and less general (because many services use mechanisms like HTTP Cookies that may not work through indirect relays) than indirection followed by redirection.

We also believe that Permafind can provide a much-needed service. Even if technically boring, Permafind might allow service mobility, which is infrequent and painful today, to become commonplace and convenient. Further, Permafind allows flexible service insertion *today* without architectural changes, allowing users to compose services such as spam filtering and virus checking. We believe there is a great need for short term solutions to mobility and service insertion, even if they are limited in scope.

However, besides meeting *present* user needs, how can Permafind foster movement towards a *future* general architecture for service portability? We believe that the appropriate end point of Internet evolution would, like in [13], have persistent service identifiers and a flexible resolution mechanism that would return metadata that contained information about which application to use (such as HTTP) and what control data (such as the desired URL) should be issued. The Permafind server, already having all the appropriate metadata, can be seen as a forerunner to this resolution service. While now it only reveals this metadata through application-level actions, the Permafind server could easily be modified to support an interface that returns the metadata directly. This interface could be invoked by a new host mechanism that replaces the standard DNS query with one that recognizes the Permafind domain and, when called to resolve such domains, asks for the metadata directly and then issues the appropriate application commands.

These two methods could coexist. Unmodified clients would go through the two-step process described earlier while modified clients would access the broader interface and get the metadata directly. Such an approach, with Permafind servers offering a broader interface to be used by modified clients, could allow the Internet to gradually evolve towards a world with persistent identifiers and flexible service invocation. We view our first deployment of Permafind as an initial step in that direction.

Moreover, such a transition would help remove an internal contradiction. This paper is about service mobility, but our approach requires users to stick with Permafind. This involves committing to a redirection service (Permafind) rather than particular application-level service providers (Gmail, etc.). While this is less noxious, it is still far from ideal. However, if we begin using modified clients that use the broader interface to request metadata, these modified clients could bypass permafind.com altogether and do a direct lookup in another resolution infrastructure. Beyond architectural cleanness, modified clients can surmount two limitations of the current Permafind: first, application names can be bound to arbitrary metadata including different types of names (*e.g.*, binding URLs to phone numbers); second, general service insertion is possible without some of the limitations imposed by unmodified clients.

More specifically, the evolution path could be (a) Permafind encourages user mobility, (b) to bypass the two-step resolution (DNS plus Permafind) and add features, users start deploying modified clients, (c) these modified clients are pre-equipped to use resolution infrastructures other than the Permafind resolver, and (d) such a new resolution infrastructure might come into being, given that there is already a set of hosts ready to use it, and the Permafind resolver is no longer a monopoly service. While this evolution story is a long-shot, we aren't aware of more credible transition paths.

We now discuss various other issues that Permafind must confront.

Security: We can conceive several threats (and possible solutions) to Permafind:

Spam: Malicious users could spam Permafind by creating names that fill the Permafind database. This can be mitigated by mechanisms like Captchas [12].

Phishing: Phishers could use Permafind addresses to hide the final destination from client browsers. As a countermeasure, we could ensure a minimum Hamming distances between Permafind names so that citibank1.permafind.com cannot be registered if citibank.permafind.com is used. Second, Permafind could disallow mappings to names (supplied by Anti-Virus companies) known to be bad. Third, instead of transparent redirection or relaying, Permafind could return a temporary web page with an explicit link to the destination URL.

Hijacking Redirects: While we argue that Redirects allow service portability, the blind following of Redirects in current browsers is also the source of many security holes. One could anticipate browsers (or application level gateways) blocking redirects in the future. As a counter, we argue that redirects are the basis of too many popular applications today to be blocked. Second, observe that the fragility of redirects is caused by browsers believing redirects sent by anyone. If Permafind or a similar resolver is considered a trusted agent then a security association (using say https) can be used to authenticate Permafind redirects.

Data and Meta Data Portability: Much of this paper has been about service *name* portability. However, there is also the issue of service *data* and *metadata* portability. Examples of data include email archives and past blogs; examples of metadata include address books and buddy lists. Porting raw data without additional semantic tags for structure is conceptually easy. Unfortunately, consider email data with an associated date tag. When moving from Gmail to Yahoo mail, a tool can easily read Bobs 2 year old stored email at Gmail and write it to Yahoo. Unfortunately, there is no way for a user tool (with-

out help from Yahoo via some externally visible API) to store old email with a specified 2 year old date.

This could be solved by introducing new APIs for each application that allow data and metadata portability. For example, if email providers decouple storage from presentation, such that users choose their storage server (*e.g.*., Amazon S3 [8]) and the email provider presents email by reading from storage using standard, externally visible standard APIs, then the data mobility problem becomes easier. To switch email presentation services, the user can make the new email presentation point to the appropriate storage server. If the user switches storage servers, a simple tool can migrate the data using standard APIs. In general, we believe that a service like Permafind must address data and metadata portability issues to make service migration easier. There is a rich set of problems in this space with both short-term and general solutions.

Provider Countermeasures: So far we have assumed that application-level service providers will stand idly by while a redirection service such as Permafind allows customers more mobility. However, there are countermeasures service providers could employ, especially to discourage relaying.

For example, suppose a provider (say Gmail) adds a one way (and secret) hash of the destination service name to the message, and this secret hash is specific to the provider Gmail. Then if the email arives back at a Gmail receiver, the gmail receiver drops the mail if the hash is incorrect. Permafind cannot compute the hash when it changes the destination email address (it is a secret hash), but leaving it unchanged condemns the packet to be dropped. A provider of a service could certainly cite security concerns for such a check while using this measure as a deterrent to services such as Permafind. Fortunately, this type of "attack" is only possible for relaying, and relaying is only needed today for email, for which relaying services such as acm.org already abound. Thus adding such a countermeasure would likely result in users crying foul. In general any countermeasure that punishes email relaying via Permafind, should also punish email relaying via Gmail and acm.org which should be too unpopular to contemplate.

For redirection, it appears that it is difficult for service providers to take counter-measures because the redirection step is not visible to the provider: the behavior seen by the provider is the same as if the user contacted the provider directly.

5 Conclusions

By allowing users to switch to best of breed services at will, service portability encourages competition amongst service providers to provide better services. Service insertion creates the further incentive of changing service intermediaries at will. While there are clearly disincentives for existing service providers (incumbents), there are incentives for new aspirants to support a service like Permafind to allow easy adoption.

In this paper, we have described the Permafind design as well as an initial prototype. The Permafind design combines two well-known mechanisms: redirection and indirection (with relaying and proxying for compatibility). Redirection at the indirection point has advantages over redirection at the old destination, or indirection (and relaying). Besides portability, the design offers a general and flexible form of service insertion for email, and a limited form of service insertion for HTTP services. While the current system has limitations, it requires no changes to existing software or infrastructure while still providing service migration and service-insertion functionality. Thus Permafind is *immediately* deployable and not just *incrementally* deployable. Further, it appears possible to gradually migrate to cleaner approaches such as [13] via this approach.

References

[1] Adjie-Winoto, et al. The design and implementation of an intentional naming system. In *ACM SOSP*, Kiawah Island, SC, Dec. 1999.

[2] H. Balakrishnan, et al. A Layered Naming Architecture for the Internet. In *ACM SIGCOMM 2004*, Portland, OR, September 2004.

[3] D. Connolly. Naming and addressing: URIs, URLs, ... W3C Architecture Document.

[4] Fielding, et al. RFC 2616 : Hypertext Transfer Protocol – HTTP/1.1.

[5] B. Frankston. DNS: A safe haven. `http://www.frankston.com/public/ESSAYS/DNSSafeHaven.asp`.

[6] M. O'Donnell. Open network handles implemented in DNS, Sep. 2002. Internet Draft, draft-odonnell-onhs-imp-dns-00.txt.

[7] M. O'Donnell. A proposal to separate Internet handles from names, Feb 2003. submitted for publication.

[8] Simple Storage Service - Amazon S3. `http://aws.amazon.com/s3`.

[9] K. Sollins. Architectural principles of uniform resource name resolution, Jan 1998. RFC 2276.

[10] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *Proc. ACM SIGCOMM*, Pittsburgh, PA, Aug. 2002.

[11] M. van Steen, F. J. Hauck, P. Homburg, and A. S. Tanenbaum. Locating objects in wide-area systems. *IEEE Communications Magazine*, 36(1):104–109, Jan. 1998.

[12] L. von Ahn, M. Blum, N. Hopper, and J. Langford. Captcha: Using hard AI problems for security. In *Eurocrypt*, 2003.

[13] M. Walfish, H. Balakrishnan, and S. Shenker. Untangling the Web from DNS. In *NSDI*, San Francisco, CA, March 2004.

The End of Internet Architecture

Timothy Roscoe
National ICT Australia, Sydney
Intel Research, Berkeley
ETH, Zürich

1 Introduction

Architecture. There's a lot of it about. Do we need it?

Recent years have seen considerable publishing activity in the area of "internet architecture". This paper steps back and asks a more radical question: is an internet architecture a good thing at all? Are we at a point in the development of distributed communications systems where the concept should be replaced by a different way of dividing up the design space?

This is not a paper about what the Right internetwork architecture should look like, but rather whether the very *idea* of a network architecture at this point in history is a help or hindrance in moving communication technology (and research in particular) forward.

We first examine critically what the role of the internet architecture is today. We argue that instead of acting as useful guide for network practitioners of all kinds (as it has in the past), the principle function the architecture performs these days is to keep the fields of "networking" and "distributed systems" separate, to the detriment of both. Put simply, the internet architecture, and more broadly the concept of a network architecture, is now *in the way*.

At the same time, trends in networking and computational hardware, and in particular in the kinds of testbeds available to researchers to validate their ideas, have made it both feasible and compelling to do research that finesses network architecture as an issue completely, and concentrates on the broader problem of building, deploying, and operating large-scale distributed applications.

Fortunately, this does not render research into "internet architecture" irrelevant, but it does call for a respinning of many of the ideas in a different context. This paper concludes by examining the new research opportunities in the area, and how they relate to tradition challenges in "architecture".

2 What has the architecture of the internet done for us lately?

It is hard to define precisely what the internet architecture is – it is a lot easier to formulate a definition of "the internet" itself, for instance. Some parts of the internet have been more or less specified (for example, protocols like TCP, and SNMP MIBs for standard components). However, unlike other networking technologies, the internet has never had a clear specification of its architecture – indeed, this may have been a prime factor in its success.

That said, some key elements today seem to be in general agreement: datagram-based connectionless service, layering of protocols, a single internet-wide protocol at the network level (the "thin waist"), placement of certain functionality (such as reliable transmission and congestion control) in the end-systems, and locating other functionality (routing, adaptation to heterogeneous networks) in the center of the network [4, 6, 24].

This architecture (if not rigid adherence to it) has had a profound effect on the internet's ability in the past to evolve into possibly the dominant networking technology today. Recently, however, the architectural view has come under increasing strain, as evidenced by deployed network technology, common practices of carriers, and debates in the research community. There is not enough space here to survey the field in detail, but we can divide the pressures on the architecture into three categories.

Pressures from within

The internet architecture is under pressure from within in two forms. The first is from required functionality which the architecture in its current form makes hard to provide, most notably security, resistance to denial-of-service attacks, and end-to-end quality-of-service, though one might also include a sound basis for charging (or, at the very least, cost recovery by ISPs).

The second is from functionality introduced into the network which doesn't fit with the principles of the architecture, such as MPLS, firewalls, network address translators, and other varieties of middlebox [25]. In many cases these developments have been a pragmatic response to requirements for extra functionality, but they invariably have consequences for the structure of the network beyond these basic requirements – for example, firewalls were introduced to provide a scoping function for network accessibility, but have resulted in a network without a systematic way to determine end-to-end connectivity for two hosts.

Pressure from above

A second, equally pragmatic response to network requirements not met by the architecture is to leave the

underlying architecture unchanged and deploy new networks above as overlays – indeed, this mimics the early design of the internet itself, and it is also in these terms that the GENI proposal is sometimes cast [3].

Ironically, by bypassing the architecture, overlays exert pressure on it in turn by making it harder for the underlying networks to perform traffic engineering without employing knowledge about what overlay a packet is part of [21]. This cross-layer peeking is, of course, somewhat contrary to the "thin waist" abstraction of the internet.

It also cuts both ways: there are well-justified calls for information to pass across the waist in the opposite direction to help overlays and other distributed applications (e.g. [17]).

Pressure from without

In addition to looking at overlays and middleboxes, it is perhaps most interesting to ask this question: architecturally, what is the internet's position with regard to other networks? Historically, the position is clear: the internet interacts with other networks by using them to carry IP packets [5]. The "thin waist" of IP makes this assimilation process easy to implement, and users of the other network gain the immediate ability to communicate any other node in the collective internet.

In practice, however, there are plenty of other relationships at work today. The various phone networks (landlines, mobile phones, SMS signalling, etc.) are actually gatewayed to the internet rather than running IP themselves, and this model is increasingly assumed for wireless sensor networks [12]. Even within the IP realm, large enterprise networks constitute significant users of bandwidth, yet do not adhere to the typical architectural principles of the internet (they typically have private address spaces, for instance).

Increasingly, the internet is viewed as one network among several, or many. Discussions of internet architecture rarely mention this shift.

Discussion

Irrespective of its past merits, it is a truism that the current internet does not conform to the traditional architecture, and there is no clear candidate architecture that captures the internet's current form. Furthermore, it is increasingly impossible to ignore the fact that the internet is just one network among many, and its current form will not allow it to encompass them as an overlay (for example, it is infeasible to run IP over sensor networks). Finally, it is unclear that the current facilities offered by the internet are where the action is: innovative communication applications like Akamai and Skype have resorted to overlays to achieve results.

None of this is news to the research community. There have been numerous proposals (indeed, entire workshops such as FDNA) for new internet architectures which address some of the problems of the internet. Given the oft-cited difficulty of evolving the internet in its current state into one with a cleaner architecture, some researchers have taken the sensible move of casting even this evolvability problem as a research challenge, and tackled it [22].

Of course, there are serious methodological problems in doing research in new network architecture. In particular, it is hard to claim success in this area without building a successful followup to the internet.

Recently in the U.S., the GENI project [1] has proposed constructing a testbed whose aims include the deployment and validation of new internet architectures. The philosophy behind GENI is articulated in Anderson et. al. [3], which also lays out two alternatives for a successful outcome of the project. The first, "purist" approach leads to a new network architecture for the next few decades. The second, "pluralist" approach results in several alternative network architectures co-existing.

However, this discussion is based on the unstated assumption that *there should be* a Network Architecture – that there is value in defining (however informally) such an architecture or architectures. The object of the present paper is to examine the opposite view: it is timely and valuable to abandon not simply the current internet architecture, but *the very idea of having one*.

3 ARCHITECTURE: WHY BOTHER?

Why have an architecture? Rather than getting bogged down in definitions ("I can't say what an architecture is, but I know it when I see it"), let's ask: What does an internet architecture hope to achieve? The traditional answers to this question [6] include: interoperability across diverse networks, easier for applications to code to, (recently) a framework for providers to compete, and finally to facilitate innovation. We should ask ourselves: does any internet architecture really address these issues?

These days, the answer seems to be "no". The internet doesn't fully handle interoperability, for example – it does not cover the space of disruption-prone networks [15] and sensor networks [12]. The uniform API of the internet has come to be a handicap to distributed application writers, who cannot exploit useful features of the underlying network, and must perform their own (often expensive) measurements to adapt to changing network conditions [10, 23]. As for competing providers, it has already been recognized that the current internet fails in this regard [7].

Since descriptions of internet architecture either refer to a non-existent present, an idealized past, or a (possibly unrealizable) future, one should ask what the role of internet architecture today is. Put another way, what does the *idea* of a network architecture do? What is the effect

of the concept being in common usage, part of "common sense"?

An alternative (and not incompatible) view of the role of network architecture is that it forms a boundary between, on the one hand "distributed systems" and "applications" research, design, and implementation, and on the other, "networking" (see figure 1). This boundary is real: it is reflected in institutions and practices as varied as large corporations (Microsoft, Google, etc. vs. Cisco, Sprint, etc.) and publishing venues (SOSP, PODC, etc. vs. SIGCOMM, IMC, etc.). Applications run in end-systems. The network carries packets.

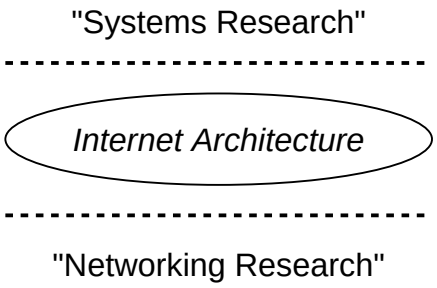


Figure 1: The Internet Architecture as a boundary between disciplines.

Of course this boundary is also rather permeable: a significant minority of researchers publish in both areas, and some venues (such as HotNets) aim at bringing together the two communities. Stated in such stark terms, the idea of the boundary looks odd, but in practice it is remarkably persistent.

For example, notice how this boundary still tends to frame the discussion: the purist vs. pluralist debate above is expressed in terms of "one or several network architectures": the purist approach is to work out what the next architecture should be by trying several out, and then build it. The pluralist approach is that there will be several network architectures in operation, and virtualization provides a way for them to share infrastructure.

To take another related example, compare the original PlanetLab paper [20], published in HotNets-I, with the "Impasse" paper [3], published two years later, in the same workshop (HotNets-III). The PlanetLab paper presents a strict superset of the vision of the Impasse paper, but from a distributed systems context[1].

The Impasse paper presents a more focussed, clearly-defined vision, but one framed entirely in networking terms – "below the line" in figure 1.

In the light of this, it is time to reassess whether the existence of a network architecture is a help or a hindrance to the general field of distributed communications, and whether it fits with the future of actual networking hardware. What would the future look like without a network architecture? What would be in its place?

4 REDEFINING NETWORKING

Modern networking hardware is very different to that available 15 years ago. It is not simply faster: there is an increasing trend toward programmability in network elements, from high-speed forwarding engines to wireless access points and radios. Programmability inevitably leads to the need to support more than one program at the same time, and so network elements increasingly support some form of *virtualization*. This trend has come at the same time as the resurgence of hardware virtualization as a building block in mainstream computing.

Virtualization has been recognized in the networking community as an enabling technology for performing basic research in network architecture. The emerging design of the GENI platform [2] can be viewed as collection of hardware "components" (computational nodes, forwarding engines, programmable radios, optical links, etc.), each of which can be sliced, or shared between different users. It is expected that most of GENI can be constructed with commodity hardware components, but using very different software.

GENI aims to be a testbed for experimenting as widely as possible with different networking technologies. Consequently, it aims (1) to mandate as little as possible about how experimenters will use a particular networking element (e.g. framing, addressing, etc.), and (2) to expose the capabilities of the hardware as much as possible to experimenters (a principle analogous to the argument for Exokernels [11]).

Experimenters are expected to acquire resources (in the form of slices of components) and build ensembles which can execute their systems. At first sight this appears to be a task of daunting complexity given the primitive building blocks available, but GENI is held together by a set of libraries and software management services which collectively enable users to compose these ensembles of components and make this a relatively straightforward process.

An important GENI deliverable is a reference implementation of a network architecture, running purely in a slice, which resembles the current internet in structure and peers with it, as an AS or collection of ASes. It is also recognized that the management services that run GENI require their own control network – initially this will be bootstrapped with an overlay[2] above the current

[1] PlanetLab has not delivered that vision, in part because it is based on deploying overlays. Overlays by themselves are incapable of providing some kinds of functionality not supported by the underlying network, for example QoS [9].

[2] An overlay is required because, ironically, the internet does not provide end-to-end connectivity between any pair of GENI nodes. For example, GENI nodes connected to Internet-2 cannot directly contact those connected to the commercial internet since Internet-2 does not peer with commercial providers.

internet, but this too is expected to move into a slice over time (see figure 2).

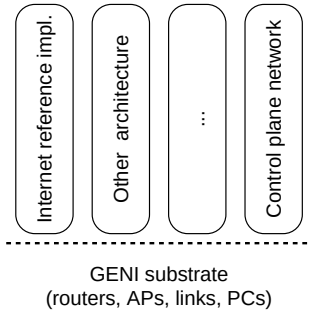


Figure 2: GENI's inversion of architecture and application.

It is a motivating goal of GENI to support research into network architectures, but note that from a broader perspective GENI inverts the traditional layering of applications and networks: the GENI substrate views experiments embodying new network architectures as applications sharing the platform, instead of defining a network architecture as a substrate for applications.

So, suppose one is a researcher who wants to deploy a new network architecture. Presumably this is because it provides some useful functions or supports some useful applications that the current internet cannot. One follows the following procedure:

1. Assemble a slice, that is, a set of virtual servers, routers, links, radios, sensors, etc.

2. Write and deploy the software implementing the network architecture to be used.

3. Write and deploy the software to peer the network with the existing internet in some way.

4. Write and deploy the newly-enabled services and applications.

From an engineering perspective, the last three steps here are all about constructing a software artifact. If the goal is to deploy distributed services that can be accessed remotely, there is no intrinsic reason to divide them up the way shown here. Calling the software in step (2) a network architecture is a rather grandiose name for what is, ultimately, just a few libraries. Carving it off into a separate unit called a network architecture is something only a network architecture researcher would care about.

Of course, we'd like code reuse, and so users deploying distributed systems atop GENI are likely to use libraries written by other parties if they are appropriate to the task at hand. Furthermore, it may make sense for some functionality to be shared between slices in the form of services accessed remotely. Over time, the use of certain libraries and services may come to be common across a wide variety of distributed systems running on the platform.

But these are purely pragmatic considerations. They do not imply anything like a "network architecture", and are unlikely to apply in all cases. Arguably, to impose a common "network architecture" on top of this substrate would be a clear violation of the end-to-end principle: ultimately, it is the application itself that can best decide how to discover, bind to, and use the resources available to it.

In this world, there is no "thin waste" – conceptually applications deal directly with physical resources sliced at as low a level of abstraction as possible, using libraries and services to make the task easier.

This not the same as saying that the GENI substrate and its management services define the "new" internet architecture (in other words, the thing that's common to all users of the hardware), for two reasons. Firstly, inasmuch as there is an architecture here at all, it is dealing with running users' code as much as shipping packets. It is not about creating a fictional boundary between two disciplines, or two types of equipment. Secondly, the structure of GENI (so far) leaves open the question of talking to other networks without mandating any common protocol, leaving the question of end-to-end connectivity an entirely application-defined issue, along the same lines as the Plutarch argument [8].

We note that this does also *not* mean that writing applications becomes much harder. It already requires tens of millions of lines of code to forward a packet from one side of the internet to the other. What changes with the dissolution of the architecture is not the complexity of this functionality, but the context in which it operates. The code now runs in application libraries and services rather than in routers - what has happened is that the total engineering space can now be carved up differently. Consequently, writing internet-like applications is no more complex than before, but writing other applications becomes possible. The substrate is no longer the barrier to innovation it is in the currently internet.

Indeed, some things may become simpler. For example, billing: each service is now using explicit resources rather than the implicit resources used in the Internet. Complex cross-provider bartering based on packet measurement isn't needed at all – if an application is sending traffic on a link, then it presumably has some code running at each end of the link, and hence it is already contracting with whoever operates each end. Carriers are now only selling low-level virtualized resources, and so have a somewhat easier operations task. At the same time, they have more opportunity to differentiate their services by innovating in the hardware they expose to users, where it is placed, and how it can be accessed.

5 REDIRECTING RESEARCH

Is this paper claiming, then, that research into architectures for the future internet and other networks is basically useless? Certainly not. The argument is that recent Internet Architecture research is not without merit, but it is currently misdirected towards the creation of one or more new "network" architectures which retain the outdated distinction between routers and end-systems.

This distinction will become increasingly at odds with reality as the PlanetLab, GENI, and Grid visions of remotely acquirable computation and forwarding resources are realized. A more productive path for the networking research community to pursue is to acknowledge that the boundary between networking and distributed systems (never more than tenuous) is dissolving, and that it is time to rethink where to draw the boundaries.

One approach is to step back and take a fresh, application-centric look. If one has the ability to create virtual machines, virtual routers, and virtual links, remotely, across diverse networks, then how does one write an application to run in this environment? What services, libraries, or other reusable components might such an application find useful?

Here is where most of the good ideas in internet architecture research can find new relevance, but they are likely to be undergo modification in the process. What those modifications are is an exciting direction for future networking and distributed systems research. We list a small selection of areas here; the reader can without doubt identify many more.

Some challenges

Routing as a library: Since applications control their own routing, a potentially rich space of application-specific routing protocols may be opened up. At the same time, many applications can of course benefit from sharing routing information and route computations. Each application is effectively setting up its own network (almost an overlay, though directly using sliced hardware rather than an existing network). There has been relatively little work in the internet arena on simultaneous routing on many overlapping graphs.

Discovery: how do applications discover and bind to a set of resources (links, routers, servers, devices)? This set is necessarily dynamic: resource availability will change due to failures, recovery, and upgrades and we can safely assume that applications will be written to adapt to changing load as well.

Unlike in traditional networking, discovery is clearly intimately tied to routing. Indeed, routing for an application might be cast as a continuous problem of discovering and acquiring the optimal set of network resources (where "optimal" is defined as some application-specific function of utility and cost).

This author has a fondness for a declarative query language approach to addressing this problem, since it neatly fuses routing and discovery [18, 19] and multi-query optimization techniques can be applied to sharing computation, but there are undoubtedly other approaches to the problem worthy of investigation.

Composition and federation: In the limit, as applications build their own networks for internal communication, how will they talk to each other? How will traffic be routed from an end system to a collection of different applications? This problem is somewhat analogous to the current problem of peering of ASes in the internet, except with a higher degree of heterogeneity to deal with, and correspondingly more freedom in implementation.

An interesting open question is whether the principal challenges in peering become harder or easier when carried out at the application layer, but note that access solutions at least can more or less directly apply techniques in systems like OCALA [16] and OASIS [13].

Operations: A final set of challenges that spring to mind with the vision of communications infrastructure in section 4 is how operators will manage the substrate. This is a worthy research challenge and is being actively pursued, but the issues are less central to the focus of this paper because they are more about individual pieces of hardware, and the distributed systems technology to remotely manage them, than about concerns which map more closely to traditional networking concerns.

Reconciling networking and distributed systems

Many of the new challenges are familiar from the field of distributed systems, but recast in such a way that they reach further down into the traditional networking stack.

In fact, the central argument in this paper has a parallel in the field of distributed systems. Traditional distributed systems research (DHTs being one good example) has tended to view the network as a black box – in particular, the network is assumed to provide connectivity between any pair of end-points [14], and quantitative differences in connectivity (bandwidth, latency, etc.) are to be recovered by the application through measurement.

6 CONCLUSION

The idea of having an architecture for a network – of carving up the space into a network and end-systems which use it – has been tremendously useful in advancing the state of the art in communications technology. However, the success of the internet has eventually resulted in this being an obstacle to radical innovation in the networking space. It is not that the architecture itself must

be fixed, but the idea itself of having a network architecture is now in the way.

Dissolving the category of network architecture allows us to move forward with the more basic problem of how to build and peer distributed applications, particularly in a future of mobile devices, sensors, smart objects, and the like.

In time, a new and useful consensus about how to build distributed communication systems may emerge, and it might then be termed an "architecture", though not necessarily of a network. Until then, we can make more progress by removing the blinkers imposed by the outdated idea of a network architecture.

7 ACKNOWLEDGEMENTS

The ideas in this paper have developed from numerous discussions on network architecture, in particular with Dave Clark, Jon Crowcroft, Kevin Fall, Steve Hand, Richard Mortier, Larry Peterson, Sylvia Ratnasamy, Andy Warfield, and John Wroclawski. I am particularly indebted to Tom Anderson for sparking the initial idea, and Scott Shenker for the invitation to present some of these concepts as a guest speaker in his Internet Architecture course at U.C. Berkeley.

REFERENCES

[1] GENI: Global Environment for Network Innovations. `http://www.geni.net/`, August 2006.

[2] T. Anderson, D. Blumenthal, D. Casey, D. Clark, D. Estrin, L. Peterson, D. Raychaudhuri, J. Rexford, S. Shenker, and J. Wroclawski. GENI: Conceptual Design Project Execution Plan. GENI Design Document GDD-06-07, January 2006. `http://www.geni.net/GDD/GDD-06-07.pdf`.

[3] T. Anderson, L. Peterson, S. Shenker, and J. Turner. Overcoming the Internet Impasse through Virtualization. *Computer*, 38(4):34–41, 2005.

[4] B. Carpenter. Architectural Principles of the Internet. Internet RFC 1958, `http://www.ietf.org/rfc/rfc1958.txt`, June 1996.

[5] V. Cerf. The Catenet Model for Internetworking. Internet Experiment Note 48, `http://www.isi.edu/in-notes/ien/ien48.txt`, July 1978.

[6] D. Clark. The design philosophy of the DARPA internet protocols. In *Proc. ACM SIGCOMM 1988*, pages 106–114, New York, NY, USA, 1988. ACM Press.

[7] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden. Tussle in cyberspace: defining tomorrow's internet. In *Proc. ACM SIGCOMM 2002*, pages 347–356, New York, NY, USA, 2002. ACM Press.

[8] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, and A. Warfield. Plutarch: An argument for network pluralism. In *Proceedings of SIGCOMM Workshop on Future Directions in Network Architecture (FDNA'03)*, pages 258–266, August 2003.

[9] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, and A. Warfield. Qos's downfall: At the bottom or not at all! In *Proceedings of SIGCOMM Workshop on Revisiting IP QoS (RIPQOS'03)*, August 2003.

[10] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl. Globally distributed content delivery. *IEEE Internet Computing*, pages 50–58, September/October 2002.

[11] D. R. Engler and M. F. Kaashoek. Exterminate all operating system abstractions. In *Proceedings of the 5th Workshop on Hot Topics in Operating Systems (HotOS-V)*, pages 78–83, Orcas Island, Washington, May 1995. IEEE Computer Society.

[12] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of the Fifth Annual International Conference on Mobile Computing and Networks (MobiCOM '99)*, Seattle, Washington, August 1999.

[13] M. J. Freedman, K. Lakshminarayanan, and D. Mazières. OASIS: Anycast for Any Service. In *Proc. 3rd USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '06)*, San Jose, CA, May 2006.

[14] M. J. Freedman, K. Lakshminarayanan, S. Rhea, and I. Stoica. Non-Transitive Connectivity and DHTs. In *Proceedings of USENIX WORLDS*, December 2005.

[15] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *Proc. ACM SIGCOMM 2004*, pages 145–158, New York, NY, USA, 2004. ACM Press.

[16] D. Joseph, J. Kannan, A. Kubota, K. Lakshminarayanan, I. Stoica, and K. Wehrle. OCALA: An Architecture for Supporting Legacy Applications over Overlays. In *Proc. 3rd USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '06)*, San Jose, CA, May 2006.

[17] R. R. Kompella, A. Greenberg, J. Rexford, A. C. Snoeren, , and J. Yates. Cross-layer visibility as a service. In *Proceedings of HotNets-IV*, November 2005.

[18] B. T. Loo, T. Condie, J. M. Hellerstein, P. Maniatis, T. Roscoe, and I. Stoica. Implementing Declarative Overlays. In *20th ACM Symposium on Operating Systems Principles (SOSP)*, 2005.

[19] B. T. Loo, J. M. Hellerstein, I. Stoica, and R. Ramakrishnan. Declarative Routing: Extensible Routing with Declarative Queries. In *Proc. ACM SIGCOMM 2005*, 2005.

[20] L. Peterson, D. Culler, T. Anderson, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *Proc. HotNets-I*, 2002.

[21] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker. On selfish routing in internet-like environments. In *Proc. ACM SIGCOMM 2003*, pages 151–162, New York, NY, USA, 2003. ACM Press.

[22] S. Ratnasamy, S. Shenker, and S. McCanne. Towards an evolvable internet architecture. In *Proc. ACM SIGCOMM 2005*, pages 313–324, New York, NY, USA, 2005. ACM Press.

[23] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. Opendht: A public dht service and its uses. In *Proc. ACM SIGCOMM 2005*, August 2005.

[24] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 2(4):277–288, 1984.

[25] M. Walfish, J. Stribling, M. Krohn, H. Balakrishnan, R. Morris, and S. Shenker. Middleboxes No Longer Considered Harmful. In *Proc. 6th Usenix OSDI*, San Francisco, CA, December 2004.

Decongestion Control

Barath Raghavan and Alex C. Snoeren
University of California, San Diego
{barath,snoeren}@cs.ucsd.edu

ABSTRACT

Congestion control is fundamental to network design. Today's networks enjoy traffic stability, performance, and fairness in large part due to the use of TCP and TCP-like congestion control protocols. In such protocols, end hosts temper their transmission rates based upon packet losses, delay, and other observations to explicitly avoid persistent congestion.

We propose an alternative view on network congestion: it may not be necessary to keep the network uncongested to achieve good performance and fairness. We argue that a protocol that relies upon greedy, high-speed transmission has the potential to achieve better performance and fairness than TCP while simultaneously guaranteeing protection against misbehaving end hosts and obviating large router buffers.

1 INTRODUCTION

One of the key problems in network design is ensuring that available capacity is fairly and efficiently shared between competing end points. Traditionally, fairness has been achieved either in the network itself using fair queuing at routers [8], or through the cooperation of end hosts using a common congestion control protocol such as TCP. Unfortunately, both approaches have significant drawbacks: fair queuing is expensive to implement, while end-host congestion control is typically far from optimal and, critically, relies on the goodwill of end hosts for success [23, 25].

While the Internet has relied upon end-host cooperation for some time, ill-conceived or intentionally-aggressive end-point behavior can drive a network without fair queuing into congestion collapse. This scenario is a classic tragedy of the commons; individual selfish behavior can drive the system to a globally pessimal state, yet there is no incentive for any user to unilaterally back off. Thus, in game-theoretic terms, the Nash equilibrium of the network congestion-control game is sub-optimal [1, 13, 14, 24, 30]. Researchers have proposed a number of router-based enforcement mechanisms to avoid congestion collapse that vary in both complexity and effectiveness: some maintain per-flow state to provide near perfect fairness [3, 8, 9, 21, 26, 27], while others simply throttle the most aggressive senders [17, 20].

We observe that most of the complexity of both fair queuing and end-host-based congestion control is due to the perceived need to avoid dropping packets: in both models, well-behaved flows should experience little or no packet loss. With the advent of efficient, high-speed erasure coding [15, 18], we argue that packet loss no longer needs to be avoided. In fact, modern coding techniques can achieve high throughput even in the face of arbitrary packet loss. Hence, we propose a novel congestion control paradigm based on fair *dropping*—as opposed to queuing—and erasure-coded data streams that is both efficient and fair.

Rather than attempting to keep the network uncongested, the goal of our proposed approach, which we term *decongestion control*, is to ensure that all available capacity is used whenever it is needed by anyone. If packet drops are not of concern, then it is straightforward to align the interests of all parties: each sender simply sends as fast as possible. If congested routers drop packets in a fair manner [19], each flow will receive its max-min fair throughput. Better yet, if flows use efficient erasure coding, they will achieve goodput almost equal to their throughput, fully utilizing network bandwidth.

Of course, there is no (non-malicious) reason for a sender to transmit faster than a path's maximum unloaded capacity; even if no other flows were present, the sender's flow would be limited by this value. The main tasks of a decongestion control protocol, then, are to enable each sender to apportion its link capacity between destinations, and to determine at what *coding* rate to transmit. While packet drops are expected, the actual drop rate (and, thus, throughput) along any given path is unknown *a priori* and will vary from destination to destination. Depending on the coding method used, it may be advantageous for senders to adjust the coding rate in response to changes in path delivery rate.

While simple in spirit, there are more intricacies and ramifications of any congestion control approach than space allows. Hence, we do not attempt to detail the full design and implementation of a decongestion control protocol here. Instead, we present a case for decongestion by enumerating the key potential benefits, briefly sketching the basics of a possible design, and concluding with a partial list of challenges that must be addressed by a real implementation.

2 BENEFITS

Decongestion controlled networks have several attractive features over and above fairness and efficiency. Because packet drops are inconsequential, routers can be simple and provisioned with smaller queues. The resulting decreased fluctuation in traffic arrival rates and predictable traffic patterns similarly simplify traffic engineering. Finally, because goodput is only dependent on packet delivery rates, the most effective malicious behavior is flooding (as opposed to timing or protocol-based attacks), which is precisely the prescribed behavior when a sender has only one flow.

2.1 Fairness and efficiency

A key challenge facing traditional end-host congestion control algorithms is determining the appropriate fair share for each flow. TCP uses an additive increase/multiplicative decrease mechanism to converge to a flow-fair allocation. Unfortunately, this allocation can take a long time to converge on high capacity and/or long-delay paths and, even in the best case, oscillates around the optimal rate. This issue is particularly acute during slow start, when the sender needs to rapidly (re-)discover an appropriate rate. While numerous modifications to TCP have been proposed to improve slow-start, they still must rely upon complex mechanisms to help TCP rapidly discover additional available capacity should it become available during congestion avoidance.

With decongestion control, in contrast, senders always transmit at the maximum available rate; fairness is ensured by appropriate dropping policies at congested routers. Should available capacity increase at any router due to, for example, the completion of a flow, the remaining flows instantaneously take advantage of the freed link resources. The ability of a flow to translate increased throughput into increased goodput of course depends on the coding mechanism employed.

Tuning the coding rate between sender and receiver is not a new class of problem, however. For example, in TCP efficiency is managed by an end-to-end control loop (*i.e.*, receive window announcements) that ensures the sending rate does not exceed the receiver's ability to consume the data. We propose to use a similar mechanism described in Section 3.1 to dynamically adjust the coding rate based on recent throughput rates. A key distinction between adjusting the coding rate and changing the transmission rate, however, is that the coding rate has no impact on other flows. Hence, changes in available capacity (and, therefore, throughput) are likely to be less frequent since traffic rates fluctuate only on flow arrival and departure events, in contrast to TCP's sawtooth which probes for additional capacity and halves its flow transmission rate upon packet loss.

Our fundamental efficiency concern is that downstream packet drops will lead to wasted capacity at upstream links, thereby decreasing the overall throughput of the network. Kelly *et al.* use the term *dead packets* to refer to packets that will eventually be dropped before reaching their destination [10]. We conjecture, however, that the slow-access/fast-core structure of the Internet may alleviate the impact of dead packets in typical topologies. Conventional wisdom states that packet loss typically occurs at access links—not in the core of the network—so most flows will be thinned out before they reach the core. Further, dead packets in the core—those that will be dropped at receivers' access links—may be inconsequential in many cases. Previous studies have shown that existing research networks (in particular, Abilene) have over-subscription factors less than 2—that is, the access links can only deliver roughly twice as much traffic as can be serviced by the transit links at each access router [28]. We hope to empirically quantify the decrease in efficiency due to dead packets in real topologies.

2.2 Simplified core infrastructure

Much of the complexity in today's routers stems from the elaborate buffering schemes necessary to ensure loss-free forwarding at line rate. In addition, TCP's sensitivity to packet reordering complicates parallelizing router switch fabrics. Adding fair queuing or similar policing mechanisms to high-speed routers even further complicates matters. By decoupling loss rate and local packet order from the end-to-end congestion control protocol, decongestion control enables significantly simpler router designs. Idealized decongestion control only requires a fair dropping mechanism, which can be efficiently implemented with a single FIFO queue [19].

In addition to their inherent complexity, a significant portion of the heat, board space, and cost of high-end routers is due to the need for large, high-speed RAM for packet buffers. Previous work has shown that erasure coding can reduce the need for queuing in the network; in particular, for networks with large numbers of flows, coding schemes can provide similar goodput with coding buffer sizes on the same order as router buffers [4]. Hence, we suspect that such a minimalistic router design would require little buffering, which, in addition to reducing cost, also decreases the variance and maximum-possible end-to-end queuing delay. While recent work has shown that smaller router buffers may suffice for large TCP flow aggregates [2], smaller router buffers make TCP more vulnerable to bursty DoS attacks [11].

We suspect that decongestion control can also simplify traffic engineering. Decongestion control in no way affects the sources or sinks of data flows, and, therefore, does not impact traffic patterns. However, due to its

inherently greedy sender behavior, some links will be driven in excess of their capacity. In contrast to today's networks, where overloaded links cause TCP goodput to plummet (due to high delays, packet loss, and timeouts), overload does not require re-engineering paths; links are equally efficient at full capacity as they are when underutilized. The net result is that engineering for overprovisioned capacity would be largely unnecessary in such a network, though backup links are still needed to cope with maintenance and failure.

2.3 Incentive compatibility

Perhaps the most compelling benefit of decongestion control is its ability to sidestep many aspects of greed and malicious behavior.

It is nearly impossible for users to unilaterally increase their goodput by injecting more packets into a network dominated by decongestion control flows, since senders are transmitting at maximum rate anyway—the most effective way for a sender to increase its goodput is to adjust its coding rate, which, as previously mentioned, has no impact on other flows. This is in contrast to TCP, whose throughput can be gamed in a number of ways, perhaps most famously by the misbehaving-receiver attacks of Savage *et al.* [23].

Decongestion control is similarly more robust to malicious behavior due to its time independence. Senders adjust coding rates based upon reported throughputs—not individual packet events—so they are not as sensitive to short-term packet behaviors as TCP. In particular, there is little opportunity to launch well-timed bursty "shrew" attacks [11]. Our goal is to reduce all attacks to bandwidth attacks: ideally, there should be nothing more effective a malicious source can do than send traffic at a high rate. Unlike shrew attacks, flooding attacks are easy to detect and defend against.

3 DESIGN

Next we consider an initial approach to designing a decongestion control protocol, Achoo. At its most basic level, Achoo sends erasure-coded packets as fast as possible between a sender and a receiver. Packets are labeled with unique, monotonically-increasing sequence numbers, and the receiver periodically acknowledges packet reception with information about the rate of reception. Ideally, all routers implement a fair dropping policy to ensure that each flow receives its fair share of link bandwidth. We conjecture, however, that enforcing fairness only at access routers would provide an acceptable level of global fairness. (Recall that TCP itself is known to be unfair to flows with varying RTTs, loss rates, etc.) The design of such a dropping mechanism is beyond the scope of this paper, but a variant of AFD [19] or a similar mechanism suffices.

Fair dropping is most important when multiple bottlenecks are involved. In the absence of fair-dropping routers, Achoo flows traversing multiple congested routers will suffer: as a flow's packets traverse each link, they compete with other flows' packets, and as a result, lose some rate. Since short flows compete at fewer links, their packets will experience a lower loss rate, and thus, yield a higher steady-state goodput. However, a fair dropping scheme prevents this path-length induced unfairness: a flow's packets are only dropped if the flow is above its max-min fair share at each router, otherwise its packets are allowed through. Thus, once a flow has been throttled to its path fair share by an upstream router, downstream routers will ensure that the remaining packets reach their destination unhindered.

Achoo's transmission behavior is controlled by two components at the sender: the decongestion controller and the transmission controller. At a high level, all data to be sent is divided into *caravans*. Each caravan consists of n fixed-size (1Kb, say) data blocks; we pick n dynamically. The role of the decongestion controller is to select the appropriate rate of transmission, rate of coding, and caravan size. The transmission controller is responsible for ensuring the delivery of each caravan of data as instructed by the decongestion controller.

3.1 Decongestion controller

The decongestion controller has three fundamental tasks: selecting the caravan size, picking the appropriate level and type of coding, and balancing transmission rates across destinations. The space of options for each of these is large: caravans can be anywhere from 1 packet to thousands of packets, coding can vary from the simple (duplicate transmission or XORs) to the complex (LT coding), and available link capacities range from tens of kilobits to many gigabits.

To select the right caravan size, the controller starts with a fixed-size caravan and begins the transmission loop. When a caravan is successfully delivered, the controller doubles the size of the next caravan. If after some fixed timeout (likely a function of the RTT) there is insufficient data in the socket buffer to fill a caravan, the caravan size is halved. In this way, the controller quickly discovers the rate at which the source is generating data.

Once the caravan size has been identified, the decongestion controller must select the type and rate of coding to use for each caravan. Many strategies can be used, each with different guarantees and tradeoffs. For now we consider a simple approach in which small caravans consist of duplicate data (ordinary redundancy) and large caravans use rateless erasure codes. This effectively trades off both the cost and latency associated with erasure coding while harnessing its strengths for larger caravans of data. Because rateless codes can be expensive

to implement, we intend to experiment with variable-rate XOR coding for modest size caravans. Senders will adjust the rate of coding in response to the successful delivery rates reported by the receivers.

The final role of the decongestion controller is to apportion the available access link capacity across flows. Each physical interface has a maximum achievable rate (which is generally far in excess of the available wide-area capacity). The job of the controller is to determine which flows can put that capacity to most effective use. Initially, the nth flow on an interface is given $1/n$ of the link capacity and the transmission rates of the other flows are decreased proportionally.

The reception rates of all flows are constantly monitored; it is possible that the current transmission rates for some of these flows are insufficient to capture available capacity (we call them *unbottlenecked*). In this case, the controller considers conducting transmission rate experiments to determine which other flows are bottlenecked and, therefore, are not making effective use of their current transmission rate.

Experiments are conducted when a flow starts a new caravan. If the reception rate for the last caravan in the flow was less than the transmission rate, the controller attempts a rate decrease and monitors the resulting end-to-end delivery rate. If a transmission rate decrease results in no decrease in delivery rate, the decrease is kept, and the newly available capacity is distributed among all unbottlenecked flows. However, if the experiment results in a goodput decrease then the previous rate is retained.

Capacity is similarly reapportioned whenever a flow finishes. Note that transmission rates are only increased for unbottlenecked flows—bottlenecked flows are not deliberately increased, but are driven slightly over their bottleneck capacity in steady state, so any increase in path capacity will be immediately reflected in delivery rate (and the flow reclassified as unbottlenecked if necessary). Of course, if a flow is not able to make use of additional rate, its receive rate will drop below the transmission rate, subsequently subjecting the now bottlenecked flow to decrease experiments. In the case where no flow is able to make effective use of the additional capacity, it may be held in reserve.

In the normal case when link access bandwidth exceeds the bottleneck capacity for all of a sender's flows, this procedure keeps each flow overdriven but converges to (just above) the lowest rate at which the maximum end-to-end goodput is achievable for each flow; in this way, the sender wastes as few resources and still maximizes its welfare. Controlling transmission rate on the order of caravans allows for bulk flows to have more stable transmission rates, since they likely send large caravans, whereas short-lived or interactive flows may need rapid rate adjustment and will have small caravans.

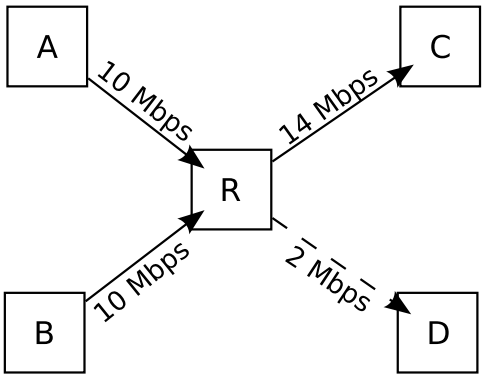


Figure 1: An example topology.

3.2 Transmission controller

The job of the transmission controller is simple: to ensure that each caravan is delivered successfully. Our approach is straightforward. Each caravan is streamed (using the rate and coding specified by the decongestion controller) until the sender receives an ACK indicating that the entire caravan has been successfully received and decoded. The transmission controller can then start sending the next caravan. More advanced designs might pipeline the transmission of caravans to eliminate the effect of network latency on inter-caravan spacing.

Interactive sessions or flows are distinguished by their sporadic data transmission. The decongestion controller uses the lack of enough data (a full caravan) as a signal to decrease caravan size: this ensures that interactive sessions transmit data in smaller units, and thus, with decreased latency. Also, since the type of coding used can depend upon the caravan size, interactive sessions can use simpler codes.

3.3 An example

To aid in an intuitive understanding of Achoo, consider the topology shown in Figure 1 depicting four end nodes, A, B, C, and D. A and B attempt to send data to C simultaneously using Achoo, which results in two flows of 10 Mbps each arriving at a router R. R implements fair dropping, so both A's flow and B's flow will achieve an end-to-end goodput of about 7 Mbps. After some time, A decides to start a flow to D, which forces it to divide its 10 Mbps between two flows. Achoo initially divides the available capacity evenly between the two flows, allowing the B-C flow to consume the remaining 9 Mbps on the R-C link. Since the link from R to D only has a capacity of 2 Mbps, if A sends at 5 Mbps to D, it will saturate the R-D link and declare the flow bottlenecked.

Conversely, the controller at A will notice that the A-C flow is unbottlenecked, and conduct bandwidth decrease experiments on the A-D flow until the A-C flow becomes bottlenecked again at 7 Mbps. In steady state, A will

overdrive its flows to both C and D in expectation of any additional capacity that may be freed up. Indeed, should the B-C flow cease, A would rapidly capture 8 Mbps for its A-C flow by conducting additional bandwidth decrease experiments on the A-D flow.

4 CHALLENGES

We consider a few of the many open questions and for each discuss the tradeoffs and issues faced by decongestion control.

4.1 What about coding overhead?

While coding provides essential functionality—resilience against loss—it also increases the end-to-end delay, packet overhead, and computational cost of decongestion control. Fortunately, different coding approaches can be used to suit the operating regime. For example, for short or interactive flows—ones with small caravans—packet duplication or simple XOR coding may yield low latency and low coding cost. As caravan size increases or there is more network contention, stronger coding schemes become necessary: flows with medium sized caravans (with sufficient computational resources) can use zero-overhead schemes such as Reed-Solomon codes, whereas large bulk flows may need to use rateless codes such as LT codes [15] or online codes [18].

4.2 What about the control channel?

Functionally, connection establishment and teardown for Achoo is the same as that for TCP. A server opens a listening socket on a particular port and establishes a new Achoo flow with each remote party sending a SYN. Similarly, once either party has decided to close the connection, they can send a close message to begin the connection teardown process. The challenge, then, is to ensure that control messages are not lost in the fray of competing data packets. For bi-directional flows, caravan ACKs can be piggybacked on coded data packets. For unidirectional flows and SYN/FINs, however, the receiver must independently determine the transmission and coding rate to use for the reverse channel.

Because the control channel contains small, independent messages, simple duplication will generally be an appropriate coding method. Determining the transmission rate, however, is more problematic. In TCP, the SYN and FIN handshakes are entirely sender-driven (a receiver only retransmits a SYN(FIN)/ACK upon receipt of a duplicate SYN/FIN). Unfortunately, with decongestion control, a single SYN/ACK packet is unlikely to be successfully delivered across a congested path. Hence, a receiver will likely need to dedicate some portion of its transmission rate for a control channel to each sender it is communicating with. A reasonable starting point is the rate currently being used by the sender to transmit its SYNs (which are piggybacked on the first data caravan for low-latency transmission of short flows). It is possible, however, that the return path is more severely congested, which would require the receiver to transmit the SYN/ACK at a faster rate.

Requiring the receiver to respond at a higher rate than the sender enables an obvious denial-of-service attack. Thus, we adopt the TCP method of requiring at least equal effort from the sender, so the receiver only increases its ACK rate in response to a commensurate increase in sender rate. Note that the transmission rate is likely far higher than the packet delivery rate at the receiver. The receiver determines the sending rate by observing the rate of change in sequence numbers of the packets it receives (each coded data packet has a unique, monotonically increasing sequence number). Implemented naively, however, the sender could lie, causing the receiver to expend undue effort.

4.3 What about unconventional routing?

In recent years researchers have proposed using alternative routing approaches such as overlay routing, intelligent multihoming, and source routing to improve performance and reliability. In such systems, flows can be redirected along different paths as traffic conditions change; additionally, in some designs, packets can be sent across multiple paths simultaneously. Achoo's relative insensitivity to instantaneous packet loss and reordering may allow for more aggressive route changes and/or multipath mechanisms than TCP would tolerate. However, decongestion, like congestion control, is path-specific, so appropriate coding and transmission rates need to be discovered after route changes.

5 RELATED WORK

The literature on congestion control and erasure coding is far too vast to adequately address here. Focusing specifically on the relationship between erasure coding and congestion control, researchers have compared ARQ schemes with FEC schemes [12] and integrated TCP and FEC [16, 22]. Naturally, FEC can be placed below, inside, or above TCP—thus, FEC can hide losses from TCP, be used to prevent retransmissions, or be applied to application-layer datagrams. Fountain codes [15, 18] famously highlighted the feasibility of non-ARQ based transport [5, 6] for broadcast and bulk data transmission. None of these schemes, however, have explored the practicality and ramifications of an entirely FEC-based congestion control on network design.

Several years ago, Davies proposed isarithmic networks in which the network is always fully utilized, just with *empties* when end hosts have no data to transmit [7]. Tracking empties proves problematic, however: just as

a token ring protocol requires a token-recovery mechanism, an isarithmic network needs some way to ensure that empties are not lost forever. Another proposal similar to ours, in that it deliberately overdrives network hosts (though not necessarily links), argues for clients of DDoS victims to increase their request rates to drown out the attackers [29], but transport flows remain congestion-controlled through TCP.

6 CONCLUSION

Given recent advances in coding techniques, we believe the time has come to consider networks that operate efficiently with high steady-state loss rates. Such a decongestion-controlled network could have simple routers, fair bandwidth allocation, low latency, stable traffic patterns, and incentive compatibility. We are currently designing and implementing Achoo and aim to quantify the potential benefits of decongestion control. Also, we note that many of the issues we are addressing have direct analogues in traditional congestion-controlled environments, so even if decongestion control is not adopted wholesale, the exercise may help us to re-evaluate tradeoffs made in the current Internet.

7 ACKNOWLEDGMENTS

We thank Rene Cruz, Bill Lin, and Scott Shenker for their thoughtful discussions. This work is funded in part through the UCSD Center for Networked Systems and a National Science Foundation Graduate Fellowship.

REFERENCES

[1] A. Akella, R. Karp, C. Papadimitrou, S. Seshan, and S. Shenker. Selfish behavior and stability of the internet: A game-theoretic analysis of TCP. In *Proceedings of ACM SIGCOMM*, 2002.

[2] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. In *Proceedings of ACM SIGCOMM*, 2004.

[3] J. C. R. Bennett and H. Zhang. WF2Q: Worst-case fair weighted fair queueing. In *Proceedings of IEEE INFOCOM*, 1996.

[4] S. Bhadra and S. Shakkottai. Looking at large networks: Coding vs. queueing. In *Proceedings of IEEE INFOCOM*, 2006.

[5] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed content delivery across adaptive overlay networks. In *Proceedings of ACM SIGCOMM*, 2002.

[6] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A Digital Fountain approach to reliable distribution of bulk data. In *Proceedings of ACM SIGCOMM*, 1998.

[7] D. W. Davies. The control of congestion in packet switching networks. In *Proceedings of ACM Problems in the optimizations of data communications systems*, 1971.

[8] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *Proceedings of ACM SIGCOMM*, 1989.

[9] S. Floyd and V. Jacobson. Link-sharing and resource management models for packet networks. *IEEE/ACM Transactions on Networking*, 3(4):365–386, 1995.

[10] T. Kelly, S. Floyd, and S. Shenker. Patterns of conjestion collapse, 2003. unpublished.

[11] A. Kuzmanovic and E. W. Knightly. Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants. In *Proceedings of ACM SIGCOMM*, 2003.

[12] S. Lin, D. J. C. Jr., and M. J. Miller. Automatic-repeat-request error-control schemes. *IEEE Communications Magazine*, 22(12), 1984.

[13] L. López, G. del Rey Almansa, S. Paquelet, and A. Fernández. A mathematical model for the TCP tragedy of the commons. *Theor. Comput. Sci.*, 343(1-2):4–26, 2005.

[14] L. López and A. Fernández. A game theoretic analysis of protocols based on fountain codes. In *Proceedings of IEEE ISCC*, 2005.

[15] M. Luby. LT codes. In *Proceedings of IEEE FOCS*, 2002.

[16] H. Lundqvist and G. Karlsson. TCP with end-to-end forward error correction. In *Proceedings of International Zurich Seminar on Communications*, 2004.

[17] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. *ACM SIGCOMM CCR*, 32(3):62–73, 2002.

[18] P. Maymounkov. Online codes. Technical Report TR2002-833, New York University, 2002.

[19] R. Pan, L. Breslau, B. Prabhakar, and S. Shenker. Approximate fairness through differential dropping. *ACM SIGCOMM CCR*, 33(2):23–39, 2003.

[20] R. Pan, B. Prabhakar, and K. Psounis. CHOKe - A stateless queue management scheme for approximating fair bandwidth allocation. In *Proceedings of IEEE INFOCOM*, 2000.

[21] A. Parekh and R. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, 1993.

[22] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM SIGCOMM CCR*, 27(2):24–36, 1997.

[23] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson. TCP congestion control with a misbehaving receiver. *ACM SIGCOMM CCR*, 29(5):71–78, 1999.

[24] S. Shenker. Making greed work in networks: A game-theoretic analysis of switch service disciplines. In *Proceedings of ACM SIGCOMM*, 1994.

[25] R. Sherwood, B. Bhattacharjee, and R. Braud. Misbehaving TCP receivers can cause Internet-wide congestion collapse. In *Proceedings of ACM CCS*, 2005.

[26] M. Shreedhar and G. Varghese. Efficient fair queuing using deficit round robin. In *Proceedings of ACM SIGCOMM*, 1995.

[27] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high speed networks. In *Proceedings of ACM SIGCOMM*, 1998.

[28] R. Vasudevan, Z. M. Mao, O. Spatscheck, and J. van der Merwe. Reval: A tool for real-time evaluation of DDoS mitigation strategies. In *Proceedings of USENIX*, 2006.

[29] M. Walfish, H. Balakrishnan, D. Karger, and S. Shenker. DoS: Fighting fire with fire. In *Proceedings of ACM HotNets*, 2005.

[30] H. Zhang, D. Towsley, and W. Gong. TCP connection game: A study on the selfish behavior of TCP users. In *Proceedings of IEEE ICNP*, 2005.

A Simple Approach to DNS DoS Mitigation

Hitesh Ballani, Paul Francis
Cornell University, Ithaca, NY

ABSTRACT

We consider DoS attacks on DNS where attackers flood the nameservers of a zone to disrupt resolution of resource records belonging to the zone and consequently, any of its sub-zones. We argue that a minor change in the caching behavior of DNS resolvers can significantly mitigate the impact of such attacks. In our proposal, DNS resolvers do not completely evict cached records whose TTL has expired; rather, such records are stored in a separate "*stale cache*". If, during the resolution of a query, a resolver does not receive *any* response from the nameservers that are responsible for authoritatively answering the query, it can use the information stored in the stale cache to answer the query. This, in effect, implies that DNS resolvers store the part of the global DNS database that has been accessed by them but use it *only* when the relevant DNS servers are unavailable. While such a change to DNS resolvers also changes DNS semantics, we show that it does not adversely impact any of the fundamental DNS characteristics such as the autonomy of zone operators and hence, is a very simple and practical candidate for alleviating the impact of DoS attacks on DNS.

1 INTRODUCTION

In the recent past, there have been many instances of flooding attacks on the Domain Name System (DNS) aimed at preventing clients from resolving resource records belonging to the zone under attack [16–19]. The frequency of such attacks on DNS can be attributed to a number of factors including, but not restricted to:

- Its pivotal role as a precursor to almost all Internet services implies that a common attack mechanism applies to a large number of services.
- Its connectionless and mostly unauthenticated mode of operation.
- The limited redundancy in nameservers and the resulting limited attack resilience [14]. Consequently, in many cases, it is easier to attack the DNS servers for a service than the actual application servers.
- Shared deployments whereby a single commercial DNS provider offers DNS services to a large number of customers are especially attractive targets due to the large attack impact [19].[1]

In response to such attacks, some of the DNS root-servers and top-level domain servers have been replicated through IP Anycast [7].

Lately, a number of research efforts have proposed new architectures for Internet's naming system. These architectures, among other things, aim to increase DNS robustness by ensuring system availability in the face of attacks. For instance, efforts arguing for a centralized DNS infrastructure [5] and a peer-to-peer based DNS infrastructure [4,13,15] represent the two extremes of this design space.

Alternatively, a complimentary tact to handle attacks on the infrastructure is to do away with the need for 100% availability. Specifically, in the case of DNS, this would entail ensuring that when the nameservers for a DNS zone are unavailable, most names in the zone can still be resolved and hence, most services in the zone are still accessible. For example, Kangasharju et. al. [9] achieve this by multicasting the global DNS database to specialized servers while Handley et. al. [6] propose a peer-to-peer design to do the same. However, we are not convinced of the need for a new dissemination mechanism to ensure DNS operation when nameservers are unavailable. In this paper we take a much more modest path and show that the need for nameserver availability in the *existing DNS framework* can be reduced simply through a minor modification in the caching behavior of DNS resolvers.

Today, DNS resolvers cache the responses they receive from nameservers to improve lookup performance and reduce lookup overhead. A resolver can use the cached responses to answer queries for a duration specified by the *time-to-live* (TTL) value associated with the response. We propose to modify the operation of resolvers such that they do not expunge cached records whose TTL value has expired. Rather, such records are evicted from the cache and stored in a separate "*stale cache*". Given a query that cannot be answered based on the cached information, resolvers today traverse down a hierarchy of DNS zones by querying the authoritative nameservers for the zone at each step. However, this resolution process fails if all the nameservers for the zone at

[1] Although it could be argued that shared deployments make attacks harder by amortizing the effort of a planned, robust server deployment.

any step of this traversal are unavailable. In such a scenario, we allow resolvers to use the information stored in their stale cache to answer the query for the unavailable zone and thus, allow the resolution process to continue.

Modifying DNS resolvers as specified above results in normal DNS operation when resolvers are able to access nameservers; only when all the nameservers for a zone do not respond to the queries from a resolver does the resolver resort to using records for the zone from its stale cache (*stale records*). This modification implies that DNS resolvers store the part of the global DNS database that has been accessed by them and use it when the relevant DNS servers are unavailable. Consequently, while attackers may be able to flood nameservers and overwhelm them, resolvers would still have the stale records to rely upon and hence, DNS availability would be less critical that it is today. We show that the stale cache can be maintained on the disk and hence, our proposal boils down to using disk-space at the resolvers to negate the impact of DoS attacks. Further, our scheme has a number of practical advantages with regards to protection against such attacks; we discuss these in section 3.1.

On the flip side, our proposal changes DNS semantics. For example, zone owners cannot expect the records served by their nameservers to be completely evicted by all resolvers within one TTL period. We analyze problems that may arise due to such semantic changes; the impact of this and other drawbacks of our scheme are discussed in section 3.2. This analysis leads us to conclude that the scheme does not adversely impact any of the fundamental DNS characteristics such as the autonomy of zone owners. Hence, we believe that the proposed resolver modification represents a very simple and practical candidate for alleviating the impact of DoS attacks on DNS.

2 A SIMPLE IDEA

2.1 DNS Resolvers Today

Clients rely on DNS primarily to map service names to the IP addresses of the corresponding servers. Typically, clients issue their queries to a local DNS resolver which maps each query to a matching resource record set (hereon simply referred to as a matching record) and returns it in the response.[2] Each record is associated with a time-to-live (TTL) value and resolvers are allowed to cache a record till its TTL expires; beyond this, the record is evicted from the cache. Given a query to resolve, a resolver executes the following actions[3]:

1. Look up the cache for a matching record. If a matching record is found, it is returned as the response.

2. If a matching record is not found in the cache, the resolver uses the DNS resolution process to obtain a matching record. This involves:

   (a) Determine the closest zone that encloses the query and has its information cached (if no such zone is cached, the enclosing zone is the root zone and the resolver resorts to contacting the DNS root-servers). For example, given an *A* record query for the name *www.cs.cornell.edu*, the resolver determines if records regarding the authoritative nameservers for the zones *.cs.cornell.edu*, or *.cornell.edu*, or *.edu* (in that order) are present in its cache.

   (b) Starting from the closest enclosing zone, traverse down the DNS zone hierarchy by querying subsequent sub-zones until the zone responsible for authoritatively answering the original query is reached or an error response from a zone's nameservers implies that the traversal cannot proceed. In either case, the resolver returns the appropriate response to the client. Also, all responses (including negative responses indicating error) during this resolution process are cached by the resolver.

3. In case the resolution process in (2.b) *fails* due to the inability of the resolver to contact all the nameservers of the relevant zone at any step of the traversal, return a response indicating the failure. Note that the term "failure" refers only to the scenario when the traversal is not completed due to the unavailability of the nameservers of a zone.

2.2 Proposed Resolver Modification

We consider DoS attacks on DNS servers where attackers flood the nameservers of a zone to disrupt the resolution of records belonging to the zone and consequently, any of its sub-zones. In general, flooding attacks aimed at denying service to clients take advantage of the skewed distribution of functionality between clients and servers. In the case of DNS, the fact that the nameservers for a zone are completely responsible for serving the zone's records and in turn, for the operation of any sub-zones implies that their availability is critical and makes them an attractive target for flooding attacks.

Changing the caching behavior of DNS resolvers so that they shoulder more of the resolution burden, especially when nameservers are unavailable, is possible within the existing DNS framework. To this effect, DNS resolvers should store the responses of the queries they

[2] Note that the matching record may not answer the query; for example, it may reflect an error condition due to which the query cannot be answered. Hence, the term "response" includes both positive and negative responses.

[3] This is a simplification of the algorithm used by resolvers but suffices for the purpose of exposition. See [10] for a more detailed version.

resolve beyond the TTL values associated with the respective responses and use stale information if all the authoritative nameservers for a zone are unavailable. Thus, the resolvers have the stale information to rely on, in case the authoritative servers for a zone are overwhelmed due to a flood of requests. More concretely, we propose the following change in the operation of DNS resolvers–

Stale Cache: Resolvers do not completely expunge cached records whose TTL value has expired. Rather, such records are evicted from the cache and stored in a separate *stale cache*. In effect, the stale cache together with the resolver cache represents the part of the global DNS database that has been accessed by the resolver.

Resolving Queries: In our proposal, the first two steps executed by a resolver when resolving a query are the same as before. Hence, given a query, the resolver attempts to respond to it based on the cached information or through the resolution process. The third step is modified as follows:

3) In case the resolution process in (2.b) fails due to the inability of the resolver to contact all the nameservers of the relevant zone at any step of the traversal, search the stale cache for the required record. If such a record is found, the resolution process in (2.b) can continue based on this stale record.

This modification implies that when (and only when) the authoritative nameservers for a zone are unavailable, the resolver can resort to responses from a previously resolved query.

Stale Cache clean-up: Existing resolvers cache the responses to the queries made during the resolution process in step (2.b). In our proposal, these responses are also used to evict the corresponding stale records from the stale cache. For example, during the resolution of the *A* record for the name *www.cs.cornell.edu*, the resolver may query the authoritative nameservers of the zone *.cornell.edu* for the authoritative nameservers of the sub-zone *.cs.cornell.edu*. When a response containing records regarding these nameservers is received, it is cached and is also used to evict any nameserver records for *.cs.cornell.edu* present in the stale cache. Note that this newly cached response will be evicted to the stale cache upon expiration of its TTL value. Also note that all responses (including negative responses) are used to evict the stale cache. For example, a NXDOMAIN response from the nameserver for *.cornell.edu* indicating that the sub-zone *.cs.cornell.edu* no longer exists will also lead to eviction of the existing nameserver record for *.cs.cornell.edu* in the stale cache. Hence, this clean-up process ensures that a record stored in the stale cache always corresponds to the latest authoritative information that the resolver received.

2.3 Stale Cache Details

From an implementation point of view, a resolver can perform steps (2.b) and (3) of the query lookup concurrently. For instance, continuing the earlier example, while the resolver queries the zone *.cornell.edu*'s nameserver for the nameservers of the subzone *.cs.cornell.edu*, it can lookup its stale cache for information regarding the nameservers for *.cs.cornell.edu*. As mentioned earlier, the information from the stale cache is used only if the resolver is unable to contact all the nameservers for *.cornell.edu* and hence, the latency of the stale cache lookup is not critical. Consequently, the stale cache can be maintained on the resolver's disk.

Given an estimated size of ≈65GB for the global DNS database [5] and the fact that resolvers maintain the stale cache on their disk, it is not far fetched to imagine that resolvers store responses for all queries that they have issued in their stale cache. In practice, we expect resolvers to assign some maximum storage space for the stale cache and utilize a popularity-based eviction algorithm (for example, LRU) when the space fills up. To come up with a back of the envelope estimate for the required storage space, we consider a week-long DNS trace collected at MIT's border router by Jung et. al. in 2001 [8]. The trace contained ≈350,000 distinct names. With an average of 100 bytes for the record(s) of each name, this would amount to 35MB of data. While DNS traffic is bound to have increased since the time this trace was collected, we can safely say that resolvers can store the responses to all queries made by them for the duration of a week with a small amount of storage space.

3 DISCUSSION

A more "clean-slate" approach to make the availability of specific nameservers less critical for the operation of Internet's naming system would be to replicate the entire DNS database at all resolvers and have authoritative nameservers only disseminate the updates for the records in their zones ([6,9] exemplify two possible approaches to achieve this). However, apart from the likelihood of the dissemination process itself being prone to attacks, any such approach could increase the total DNS overhead many times over, especially in the face of the use of DNS for load balancing purposes.

On a more general note, while most of us agree that DNS is afflicted by a few problems, we think that a majority of them can be attributed to misconfigurations, improper implementations, violations of best current practices, or even a lack of motivation to address them and not to major architectural flaws. For example, problems regarding high lookup latency can mostly be attributed to misconfigurations (i.e. broken and inconsistent delegations) [13] and the long timeouts used by resolvers

in case of errors [12]. Consequently, despite a number of proposals arguing to the contrary [4–6,9,13,15], we do not see a pressing need for an architectural change. Guided by this observation, our proposal represents an exercise in showing how minor operational modifications can address DNS problems; specifically, modifying the caching behavior of DNS resolvers can reduce the impact of flooding attacks on DNS.

In the rest of this section we discuss the advantages of the proposed modification and a few possible objections to it.

3.1 Pros

DNS Robustness. The proposed modification ensures that resolvers can respond to queries for a zone even if the zone's authoritative nameservers are unavailable, assuming that the resolver has queried the zone at some point in past and the previous response is present in the stale cache. DNS's hierarchical structure entails that almost all modified resolvers would have information for zones higher up in the hierarchy, such the DNS root-zone and the top-level zones, stored in their stale cache. Hence, the proposal would significantly reduce the impact of DoS attacks on such zones.

As a matter of fact, popularity of DNS names follows a zipf-like distribution [8]. Consequently, stale responses for a large fraction of queries to be issued by a resolver in the near future should already be present in the resolver's stale cache. Thus, having the stale cache in place insures the resolver (and its clients) from DoS attacks against DNS nameservers since a large fraction of queries can, if needed, be answered based on the records in the stale cache.

Simplicity. The biggest argument in favor of this proposal as a means of increasing DNS robustness is its simplicity. The proposed scheme:

- *Does not change the basic protocol operation and infrastructure*; only the caching behavior of resolvers is modified.
- *Does not impose any load on DNS*, since it does not involve any extra queries being generated.
- *Does not impact the latency of query resolution*, since the stale cache is utilized only when the query resolution fails.

Incremental Deployment. Any single resolver can adopt the modifications proposed in this paper and achieve significant protection from attacks against the DNS servers it and its clients access. Hence, the proposal can be incrementally deployed.

Motivation for Deployment. Modifying a resolver is beneficial for the clients being served by the it since the resolver can resolve queries for zones that have been accessed by it in the past even if the nameservers for the zones are not available. Hence, there is motivation for the resolver operators to switch to the modified resolver.

3.2 Objections

DNS caching semantics and the possibility of obsolete information being used. The biggest objection against the proposed modification is that it changes the semantics of DNS caching. With the current DNS specifications, a zone operator can expect the records served by the zone's authoritative nameservers to be completely expunged by resolvers within TTL seconds.[4] With our proposal, such records would be evicted to the stale cache. The problem with such an approach is best explained through an example. Let's consider a zone whose records have been updated. Also, consider a resolver that has accessed the zone but not since the update and so, its has the zone's obsolete records in its stale cache. We don't place any bound on the time for which the records can be kept in the stale cache. So, if the resolver needs to resolve a query for the zone at a time when all the zone's authoritative nameservers are unreachable, it would resort to using the obsolete records present in the stale cache.

The problematic scenario described above arises only when all the authoritative nameservers for a zone are unavailable. In such a scenario, existing resolvers would fail to resolve any queries pertaining to the zone or any of its sub-zones (assuming that the records for the sub-zones are not present in the resolver cache). For the modified resolvers, if the resolver has not accessed the zone since the zone's records were last updated, it would use obsolete information. While this is far from perfect, the small possibility of obsolete information being used seems like a small price to pay for the robustness offered by having a stale cache in place. Also, the possibility of a resolver using obsolete information for a zone is much less for zones that the resolver frequently accesses.

Further, resolvers may choose to apply the modified caching scheme to infrastructure records only. Infrastructure records, as defined by [11], refer to records used to navigate across delegations between zones and include the *NS* records (and the corresponding *A* records) for zones. Past studies show that such records change very infrequently [6,11] and hence, this would further reduce the possibility of resolvers using obsolete information while still providing a large robustness gain.

Attackers attempting to force the use of obsolete information. Apart from the possibility of obsolete data being used, there is also the possibility of attackers taking advantage of the stale cache maintained by resolvers to

[4] In practice, zone operators need to be more flexible due to a large number of misbehaving resolvers that disregard TTL values and use expired records even though the nameservers for a zone are available [22].

force the use of obsolete data. Attackers may keep track of updates to the records of a zone and start flooding the authoritative nameservers for the zone as soon as some of the records are updated. If the attack overwhelms the zone's nameservers, resolvers trying to resolve the zone's records would rely on the obsolete data stored in their stale cache. In effect, attackers can now flood the nameservers for a zone in order to delay the propagation of updates to the zone's records for the duration of the attack. While this is certainly a possibility, we have not been able to imagine scenarios where this would be worse than not being able to access the zone at all (which is the case with the status quo).

Autonomy for zone operators. A related concern is that the proposed modification would seem to move autonomy away from zone operators to resolver operators. Allowing resolvers to store records after their TTL value has expired suggests that zone operators do not control the access to their sub-zones; for instance, they could not kill off their sub-zones when they wish to.

However, this is not the case. The fact that we don't modify DNS's hierarchical resolution process implies that resolvers still need to go through the nameservers for a zone in order to access its sub-zones and hence, the autonomy of zone operators is not affected. For instance, let's assume that the operator for the zone *.cornell.edu* needs to kill off the sub-zone *.cs.cornell.edu*. Typically, this would involve zone *.cornell.edu*'s operator configuring the zone's authoritative nameservers to respond to any queries regarding *.cs.cornell.edu* with a NXDOMAIN, implying that no such domain exists. Consequently, a resolver trying to resolve a query like the *A* record for *www.cs.cornell.edu* by traversing down the DNS zone hierarchy would receive a NXDOMAIN response from one of the nameservers for *.cornell.edu* and would forward this to the client that originated the query. Further, this response would be cached and eventually be evicted to the stale cache. Thus, if there are any such future queries at a time when all nameservers for *.cornell.edu* are unavailable, the resolver would still return a NXDOMAIN response.

Resolution latency in the face of an attack. In our proposal, if a resolver is unable to reach the authoritative nameservers of a zone, it resorts to using the zone's records in the stale cache. Consequently, the resolver must query each of the nameservers for the zone, wait for the query to timeout (and possibly retry) before it can use the stale cache. With the current timeout values used by resolvers, this would entail a high lookup latency in the face of attacks (i.e. when the nameservers for a zone are unavailable). For example, the default configuration for the BIND8 resolver [21] involves sending queries to each nameserver for 30 seconds with an exponentially increasing period between consecutive retries. So, clients accessing a zone with two authoritative nameservers at a time when both of them are unavailable would need to wait for 60 seconds before receiving a reply. However, most resolvers allow the retry and timeout values to be configured and hence, the lookup latency problem can be solved by using aggressive values for these timers. As a matter of fact, past work has already suggested that these timer values are major contributors to the high lookup latency when errors are encountered [12].

DoS'ing the application servers. The proposed modification does not reduce the vulnerability of nameservers to DoS attacks. Consequently, attackers can still flood them so that they are unable to serve (and update) the records of the corresponding zones. Rather, the modification makes the availability of DNS nameservers less critical and hence, significantly reduces the impact of DoS attacks on DNS.

Further, the proposal does not address the general DoS problem and attackers can deny service to clients by attacking the application servers instead of the corresponding DNS nameservers. As a matter of fact, a flooding attack that chokes the network bottleneck for a zone's nameservers is also likely to hamper the availability of the zone's application servers. In such a scenario, there isn't much value to being able to resolve the names for the application servers since clients would not be able to reach them anyway.[5] In effect, this concern boils down to how common is it for application servers and their nameservers to share a network bottleneck. We intend to measure this for nameservers on the Internet as part of our future work.

Interaction with DNSSec. The proposal does not have any harmful interactions with or implications for DNSSec. In case the resolver cannot reach the nameservers of a zone and relies on the corresponding records in the stale cache, the records ought to be classified as "Undetermined" by the resolver.[6] Hence, any DNSSec policies expressed by the resolver operator for undetermined records naturally apply to the stale records.

4 Related Work

A number of recent efforts [4–6,13,15] have proposed new architectures for the next generation Internet naming system that address DNS's performance and robustness problems. Balakrishnan et. al. [1] propose to replace the hierarchical DNS (and URL) namespace with flat identifiers. We show that a minor operational change to resolvers in the *existing DNS framework* can significantly mitigate the impact of DoS attacks on DNS.

[5] Note that there is still a lot of value to being able to access the sub-zones when a zone's nameservers are being fboded. For example, being able to access the rest of the name system when the root-servers are being fboded.

[6] Undetermined records correspond to records resulting from a non-DNSSec lookup [20].

Pappas et. al. [11] argue for the use of long TTL values for infrastructure DNS records as a means of alleviating the impact of DoS attacks on DNS. We share with their proposal the basic notion of using records already present in the resolver cache for a longer period. While our proposal involves changing the caching behavior of resolvers, using longer TTL values for a zone's records involves a minor configuration change at the zone's nameservers and hence, does not necessitate any software update. However, using long TTL values does not completely offset the impact of DoS attacks (since a fraction of the records still expire at any given time) and makes it harder for operators to update their records.

Cohen and Kaplan [3] propose the use of stale DNS records for improving DNS performance. This involves fetching data based on the stale records and issuing a DNS query to refresh the stale record concurrently. As a contrast, we argue for the use of a zone's stale records only in case all nameservers for the zone are unavailable. CoDNS [12] is a cooperative DNS lookup service designed to alleviate client-side DNS problems. We share with their proposal the notion of client-side (i.e. resolver-side) changes to address DNS problems. While CoDNS involves resolvers co-operating amongst each other to mask resolver-side issues, we propose that resolvers use their disk-space to insure themselves (and their clients) against DoS attacks on DNS.

5 Future Work

This paper presents a very simple modification to the caching behavior of DNS resolvers. While the impact of changing a resolver on the resolver's ability to cope with DoS attacks on nameservers is pretty obvious, we would like to use real-world traces to quantify this impact. To this effect, we are in the process of obtaining traces collected at DNS resolvers and aim to determine the impact of different kind of attacks. For example, assuming an attack that takes down the DNS root-servers, how many queries for a typical resolver will fail, how many will use obsolete information (in case the nameservers for one of the TLDs changes during the attack), and how many will benefit from having the stale cache in place?

We also aim to implement this modification into common DNS resolvers (for example, BIND, DBJDNS, etc.) or even as an add-on to the CoDNS resolution service [12] running on PlanetLab [2]. Apart from clearing up the implementation issues, such an exercise would help us analyze the advantages of maintaining a stale cache in the face of actual attacks (which occur frequently enough to make this exercise worthwhile!).

Acknowledgements

We would like to thank Paul Vixie at ISC for helpful discussions on why this proposal should "not" be incorporated in DNS resolvers; most of the objections discussed in section 3.2 arose during exchanges with him.

References

[1] Balakrishnan, H., Lakshminarayanan, K., Ratnasamy, S., Shenker, S., Stoica, I., and Walfish, M. A Layered Naming Architecture for the Internet. In *Proc. of ACM SIGCOMM* (2004).

[2] Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., and Bowman, M. PlanetLab: An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review 33*, 3 (July 2003).

[3] Cohen, E., and Kaplan, H. Proactive caching of dns records: Addressing a performance bottleneck. In *Proc. of Symposium on Applications and the Internet* (2001).

[4] Cox, R., Muthitacharoen, A., and Morris, R. T. Serving DNS using a Peer-to-Peer Lookup Service. In *Proc. of IPTPS* (2002).

[5] Deegan, T., Crowcroft, J., and Warfield, A. The Main Name System: An Exercise in Centralized Computing. *SIGCOMM Comput. Commun. Rev. 35*, 5 (2005).

[6] Handley, M., and Greenhalgh, A. The Case for Pushing DNS. In *Proc. of Hotnets-IV* (2005).

[7] Hardy, T. RFC 3258 - Distributing Authoritative Name Servers via Shared Unicast Addresses, April 2002.

[8] Jung, J., Sit, E., Balakrishnan, H., and Morris, R. DNS performance and the effectiveness of caching. *IEEE/ACM Trans. Netw. 10*, 5 (2002).

[9] Kangasharju, J., and Ross, K. W. A Replicated Architecture for the Domain Name System. In *Proc. of INFOCOM* (2000).

[10] Mockapetris, P. RFC 1035, DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION, Nov 1987.

[11] Pappas, V., Zhang, B., Osterweil, E., Massey, D., and Zhang, L. Improving DNS Service Availability by Using Long TTLs. draft-pappas-dnsop-long-ttl-02, June 2006.

[12] Park, K., Pai, V., Peterson, L., and Wang, Z. CoDNS: Improving DNS Performance and Reliability via Cooperative Lookups. In *Proc. of USENIX OSDI* (2004).

[13] Ramasubramanian, V., and Sirer, E. G. The Design and Implementation of a Next Generation Name Service for the Internet. In *Proc of ACM SIGCOMM* (2004).

[14] Ramasubramanian, V., and Sirer, E. G. Perils of Transitive Trust in the Domain Name System. In *Proc. of ACM SIGCOMM IMC* (2005).

[15] Theimer, M., and Jones, M. B. Overlook: Scalable name service on an overlay network. In *Proc. of ICDCS* (2002).

[16] Microsoft DDoS Attack, NetworkWorld, Jan 2001. `http://www.networkworld.com/news/2001/0125mshacked.html`.

[17] Root Server DDoS Attack, RIPE Mail Archive, Nov 2002. `https://www.ripe.net/ripe/maillists/archives/eof-list/2002/msg00009.html`.

[18] Akamai DDoS Attack, Internet Security News, Jun 2004. `http://www.landfield.com/isn/mail-archive/2004/Jun/0088.html`.

[19] UltrDNS DDoS Attack, Washington Post, May 2005. `http://blog.washingtonpost.com/securityfix/2006/05/blue_security_surrenders_but_s.html`.

[20] CISCO DNSSEC page, Aug 2006. `http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_7-2/dnssec.html`.

[21] Internet Systems Consortium, Aug 2006. `http://www.isc.org/`.

[22] SLASHDOT: Providers Ignoring DNS TTL?, Aug 2006. `http://ask.slashdot.org/article.pl?sid=05/04/18/198259&tid=95&tid=128&tid=4`.

SPACE: Secure Protocol for Address-Book based Connection Establishment

Ganesh Ananthanarayanan[†,1], Ramarathnam Venkatesan[†,‡,2,3], Prasad Naldurg[†,4], Sean Blagsvedt[†,5] and Adithya Hemakumar[†,5]

[†]Microsoft Research India, [‡]Microsoft Research Redmond
{ganeshan, venkie, prasadn, seanb, t-adith}@microsoft.com

Abstract

We present SPACE an application-level protocol for secure automatic ad-hoc connection-establishment between two devices based on their address book entries. Our protocol is based on the simple premise that if two people have each others contact details in their address books, they probably know and trust each other in some limited way and this can form a basis for a trust relationship between their devices, without additional user intervention. We show how our protocol is resistant to specific security attacks and can accommodate for privacy concerns. Existing connection-establishment protocols for Bluetooth and IEEE 802.11 have known security flaws, and can be compromised using well-known techniques and off-the-shelf hardware. In addition, these protocols require explicit user intervention, like entering a passkey. We believe that these factors have directly impacted the widespread application of ad-hoc networking in the context of mobile phones and other consumer devices.

1 INTRODUCTION

The last decade has seen a huge increase in the number of consumer devices with integrated wireless communications (Wireless Local Area Network WLAN) such as Bluetooth and Wi-Fi that enable users to connect to other nearby devices in a direct peer-to-peer fashion. The potential benefits of these direct connections were once heralded as significant allowing users to easily share resources such as files, computational power and network connectivity, and engage in collaborative applications. With the possible exception of the most recent generation of dedicated portable videogame players, we feel that the potential of peer-to-peer ad-hoc networking has not yet been achieved, especially on mobile phones and laptop computers. We identify two particularly weak points in the connection establishment process viz., security concerns and user inconvenience.

Users are wary about unauthorized and potentially malicious access to their devices that could compromise the privacy of their data. Existing connection establishment mechanisms in Bluetooth and IEEE 802.11x can be compromised using well-known techniques and off-the-shelf hardware [8, 11] and incidents of Bluetooth-borne viruses have been reported. In addition, the connection establishment techniques are cumbersome and often require explicit user intervention. In most Bluetooth implementations, the user must enter a passkey [12] for each connection.

We model trust relationships in such scenarios on real-life relationships among users and devise an automatic and secure application-level protocol for ad hoc connection establishment. Our guiding insight is that if two people have each others contact details (e.g. phone number, email address) in their address books, it means that they are more willing to trust each other in some limited way, and this can form the basis for a trust relationship for their respective devices, without additional user intervention. We believe that if the problem of trusted automatic connection establishment can be satisfactorily solved, this could provide the basis for many other exciting applications such as file and connection sharing and a connected world where our devices are smart enough recognize the people around us that we know, just as humans. Subsequently, additional security and access controls can further improve the security of this connection.

To this end, we present *SPACE*, an application-level protocol for automatic ad hoc connection establishment between two devices based on their address book entries. *SPACE* has two phases. In the first phase - *Scan Phase*, the devices broadcast a keyed, cryptographic hash of the users contact details and also check the presence of the address corresponding to an incoming hash in their address book. In the second phase - *Authentication Phase*, they establish a shared secret key between them via a secure external network channel, which is resilient to impersonation, such as the exchange of SMS or email messages, and subsequently authenticate each other using an encrypted nonce exchange.

We make two important contributions: (1) we propose a protocol for an automatic ad hoc connection establishment between mobile devices based on a novel model of trust that

[1]Hardware, Communications and Systems Group
[2]Cryptography, Security and Algorithms Group
[3]Cryptography and Anti-Piracy Group
[4]Rigorous Software Engineering Group
[5]Advanced Development and Prototyping Group

is founded on contact details, and (2) we solve the problem of impersonation in this setting by performing a key-agreement over an external network that provides a reasonable guarantee of identity pertinent to the specific contact details that are being exchanged (e.g. the key agreement for mobile numbers occurs via SMS). We analyze the security of our protocol and show that it is resistant to impersonation attacks unlike Bluetooth [4, 12] and IEEE 802.11b [4].

2 TRUST MODEL

Devising trust models for ad hoc networks is still an open and challenging area of research. Ad hoc networks can be broadly classified in to two categories based on their method of trust establishment [2, 13]. *Managed* ad hoc networks are based on the assumption of the existence of a central authority to distribute and verify *certificates*. In *Pure* ad hoc networks trust is dynamic and is based on reputation and recommendations from peers in the network. While the models for managed ad hoc networks require a central authority for trust establishment, their counterparts for pure ad hoc networks assume the absence of any initial knowledge vis-a-vis trust. We believe that the assumptions of a central authority and absence of any prior information significantly reduces the utility and effectiveness of peer-to-peer ad hoc connectivity.

We try to model trust in peer-to-peer ad hoc networks based on real-life relationships. Popular forms of peer-to-peer ad hoc connectivity are generally based on a *social* model of trust, e.g. share photos via Bluetooth with devices of people who you know personally. As a natural extension, people are likely to store the contact details (like phone numbers) of people who they know and we leverage this as a basis for trust in *SPACE*. Devices that have each other's contact details in their address books are likely to trust each other, at least in some limited form, and should be able to automatically detect and connect to each other.

Our trust model has the characteristics that are a hybrid of trust models for managed and pure ad hoc networks. While we assume no central authority for trust management, we piggyback on the existing and popular mechanism of exchange of contact details for our *certificate* distribution. The significant advantage of our trust model is its ready deploy ability unlike existing security mechanisms for peer-to-peer ad hoc networks (e.g. there is no viable distribution mechanism for Personal Identification Numbers (PIN) in Bluetooth).

Our trust model raises two relevant questions - (1) people may not trust everyone in their address books, and (2) a user's address book may not contain his entire list of trusted people. The former can be addressed easily by adding an extra field in the address book that specifically marks whether a contact can be trusted for peer-to-peer ad hoc connectivity. The latter problem falls in the category of the larger unsolved problem of *certificate* distribution for peer-to-peer ad hoc networks. We propose a partial solution to this by utilizing real-life relationships for trust establishment.

3 PROTOCOL DESCRIPTION

In this section we describe *SPACE* protocol. Consider two users Alice and Bob with address books $AD_a = \{a_1 ... a_m\}$ and $AD_b = \{b_1 ... b_n\}$ where a_i and b_i represent the individual addresses, and m and n are the sizes of the address books. Let their self-addresses be s_a and s_b respectively. We will refer to the devices as Alice and Bob in the rest of the paper. The protocol has two phases.

3.1 Scan Phase

In *Scan Phase*, Alice and Bob verify the presence of each others contact details in their respective address books. The phase starts with Alice and Bob computing a keyed cryptographic hash $H_{ka}(s_a)$ and $H_{kb}(s_b)$ of their contact details, using randomly generated keys k_a and k_b, and broadcasting it along with the key. Fig 1 represents a part of the broadcast that happens during Scan Phase.

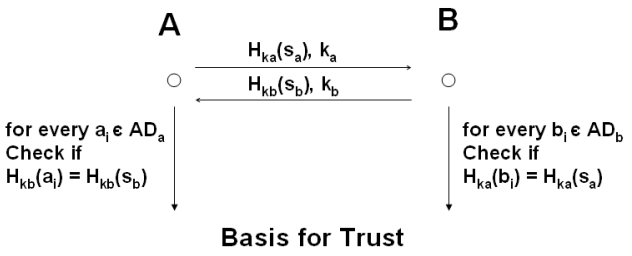


Figure 1. Scan Phase of the *SPACE* protocol

H can be any pre-image resistant and collision-free hash function like SHA_1. Since the hash keys k_a and k_b are randomly and periodically changed, the hash values are different for every broadcast. This prevents the association of the hash value itself as an identity of a device, which may raise privacy concerns.

On receiving this message, Alice and Bob scan their address books to check if the hash of any of the entries in their address books matches the value they have obtained. By assumption the hash value cannot be inverted to get the contact detail in feasible amount of time and hence this guarantees privacy. Also, we can increase the search space by including some additional information with the address before computing the hash value to give a better guarantee of the hash value being non-invertible. But it is to be noted that an adversary can partially invert the hash and obtain a few bits out of it (e.g. obtain the bits corresponding to the area code in the hash and hence violate privacy). While this is not a problem currently with hash functions like SHA_1, we can provide a complete guarantee against inversion by adding a random key to the hash value and re-hashing it.

If Alice and Bob find an address in their address books whose hash matches the value they obtained from the other party, they have a basis for believing that the other person is present in their address book and hence can be trusted. If

Alice or Bob cannot find such an address in their address books, the protocol halts.

3.2 Authentication Phase

In most real-world scenarios, the address books contain either an e-mail address or a phone number. Since the confidentiality of these contact details cannot be guaranteed, this can result in malicious users impersonating their contact details. Consider the scenario where there is a malicious device Ian that wants to connect to Alice. If Ian can find an address $i \in AD_a$, then he can claim his address to be i and hence would get authenticated by Alice because Alice has no means of ascertaining the veracity of Ians claim. This phase deals with the problem of impersonation where the device has no method of verifying the veracity of the address claimed by the other device in the Scan Phase. While Scan Phase identifies the nodes present in a device's WLAN, Authentication Phase ensures that the identities of those nodes are authentic.

We introduce a one-time key agreement step in *SPACE*. Here Alice and Bob agree on a shared key between them via a secure external network channel, which is relatively safe as a way to detect who is sending it - if not as secure as a method for exchange of information such as the exchange of SMS. With devices increasingly becoming multi-homed, we believe that the assumption of an external network channel is reasonable. Fig 2 describes the Elliptic curve based Diffie Hellman key agreement protocol. P is a point on the elliptic curve [3, 7] and is publicly known. Note that Alice and Bob arrive at the same key ($a.b.P = b.a.P$) *without explicitly transmitting the key at any point in time*. The key agreement protocol is based on the hardness of the Elliptic Curve Discrete Logarithm Problem [6] (i.e.) given $a.P$ and P, it is *computationally hard* to find a. Elliptic curve based key agreement was chosen because of its small key size and relatively low overhead in the cryptographic operations [3, 7]. In the first connection instance between two devices, they securely agree to a key between them and use this key for authentication. Note that *SPACE* performs key *agreement* and not key *exchange*, and hence is not susceptible to passive interception of messages.

A point to note is that nodes impersonating their identities (contact details) will not be able to perform the key agreement successfully and will not have the correct key values. This holds true in case of the common address book entries like phone numbers and e-mail addresses. If the key agreement is performed via SMS or mail exchange, then a person impersonating his contact detail will not receive the messages and hence unable to compute the key.

The key is unique for every device pair. Every device has a key table KT with fields contact address c and key k. This table is initially empty and can grow to a maximum length of the size of the address book.

Now Alice and Bob present a nonce-challenge to validate each other. Alice generates a random nonce N_a, encrypts with the key K_{ab}, the key corresponding to the contact address s_b in Alices key table KT_a, and sends it to Bob over the WLAN. Bob uses his key K_{ba} to decrypt the message and obtain N_a. If Alice and Bob had successfully performed their key agreement then $K_{ab} = K_{ba}$. Bob concatenates N_a along with his own nonce N_b, encrypts it with K_{ba} and sends it to Alice. Alice checks the correctness of N_a and sends back N_b for Bobs verification. If Alice and Bob decrypt and send the right nonce values back to each other, they can conclude that there is no impersonation.

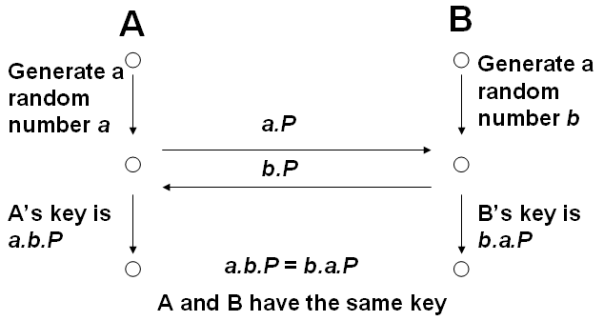


Figure 2. Elliptic Curve Diffie Hellman Key Agreement

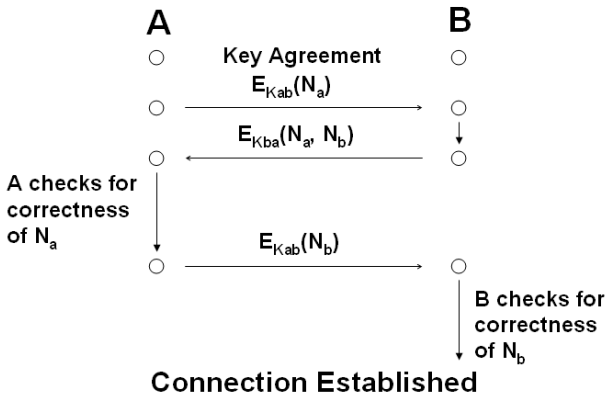


Figure 3. Authentication Phase of the *SPACE* protocol

4 Security Analysis

In this section, we present our analysis of the *SPACE* protocol described in Section 2. We do not present a formal proof of security protocols but instead outline the security assumptions and present our arguments. In the Scan Phase, the preimage resistance of the hash functions is necessary for privacy concerns. In the Authentication Phase, the messages in the key agreement protocol need to be authenticated to be tamper-resistant when necessary and the encryption function used for the nonce challenge needs to be secure. Hence, the security of the protocol primarily depends on the following factors:

1. The security of the standard symmetric key encryption algorithms, such as AES.

2. The keys agreed upon using the public key exchange protocol are indistinguishable from those obtained from a cryptographic pseudorandom number sequence; otherwise an adversary may guess the some bits of the information.

3. Impersonation and message tampering does not happen in the secure channel over which the key agreement happens.

One should choose the parameters such as the key lengths as per the existing standards for public and private key cryptographic primitives.

The final item is the assumption that impersonation and data tampering is not possible in the network over which the key agreement is performed. Passive interception of packets without tampering has no bearing on the security of our protocol (see Section 4.4). In our actual implementation and testing, we used the Short Messaging Service (SMS) over the cellular network for key agreement. The Universal Mobile Telecommunication System (UMTS) has adopted an enhanced authentication and key agreement protocol for 3G communications [1] which includes data confidentiality and user identity privacy [9]. Hence, one may assume that it is sufficiently hard to impersonate and tamper with data in the cellular network.

We now examine some specific attacks which have exposed weaknesses in existing protocols [4, 12], and discuss their impact on our protocol.

4.1 Man-In-the-Middle Attacks

In this attack, a malicious user Ian intrudes into a conversation between Alice and Bob in the WLAN. Ian obtains the messages from Alice and forwards it to Bob and vice versa and makes them believe that they are talking to each other.

In our protocol, in the Scan Phase, Ian obtains the hash of Alices number and forwards it to Bob and does the same with the hash of Bobs number. Hence now Alice and Bob can establish a basis for trust between them even though they are not talking to each other directly.

However in the Authentication Phase, Ian cannot participate in the key agreement protocol and cannot compute the shared key. When he receives an encrypted challenge in the Authentication Phase, he cannot decrypt it to obtain the nonce. In effect Ian can act as a wire in between or scuttle the protocol by tampering with the messages before forwarding it. Therefore, Alice and Bob will not end up connecting to each other. This is futile for Ian and hence this is not a security vulnerability.

Replay Attack: Consider a malicious user Ian who can record and replay the broadcast from Alice to Bob. Bob will be able to find a match in his address book and conclude that there is a basis for believing Ian. Ian can ignore the step of finding a match in his address book. However, because of the Authentication Phase, Ian will not be able to authenticate successfully. If Alice and Bob are connecting for the first time, Ian will not be able to obtain the key agreement messages and will not be able to compute the key. If Bob already has an entry for Alice in his key table, it is infeasible for Ian to compute the key.

4.2 Contact Detail Compromise

In common scenarios, safeguarding ones contact detail (e.g. phone number, e-mail address) is very difficult and is highly likely to be known to people who need not be trustworthy. But this is not a security vulnerability in *SPACE* because it is based on the notion of whether a device has the other devices contact detail in its address book. As long as the device does not engage in a secret key agreement protocol with some user who is malicious (e.g. it trusts that people in its address book are non-malicious), it is secure.

4.3 Denial of Service

A mobile device is especially vulnerable to denial of service attacks as it has limited energy (battery) resources. With respect to our protocol, a determined attacker can effectively mount denial of service attacks. An attacker Ian can impersonate himself to have any address in Bobs contact list AD_b. Hence he will pass through the Scan Phase but will fail in the Authentication Phase.

In the Scan Phase, Bob has a computation cost involved in hashing all the addresses in his address book. This cost can be reduced by precomputing and storing the hashes once and reusing them. Since, we use keyed hashes, Bob can periodically refresh the keys and re-compute the hash values. If Ian impersonates someone for whom Bob already has a key table entry, then Bob will be forced to perform one encryption operation for its nonce and one decryption operation to check for the correctness of the nonce received from Ian. If Ian impersonates someone with whom Bob has never connected before then he would make Bob perform a useless key agreement process which would make him incur computation as well as communication costs. The computation overhead can be reduced by using symmetric key based cryptographic methods and hence significantly reduce the costs associated with the encryption and decryption operations. Elliptic curve based key agreement protocols significantly reduce the size of the keys in the agreement process and decreases the communication costs.

4.4 Key Agreement Interception

The security of the messages exchanged during the key agreement phase is very critical. The key agreement protocol is resistant towards passive interception of the messages because of the hardness of the Elliptic Curve Discrete Logarithm Problem [11]. The attacker Ian will have to perform a man-in-the-middle attack during the key agreement process. He can tamper with the messages between Alice and Bob, resulting in both having a modified unequal key. Subsequently, he can impersonate himself as either of Bob or Alice

and connect with the other device. Note that Alice and Bob will not be able to talk to each other in this case. Fig 4 shows how Alice and Bob end up with different keys that only Ian is aware of, in the Elliptic Curve Diffie-Hellman Key Agreement protocol. After the key exchange process, Alice and Bob will not be able to authenticate each other as they do not have the same key.

In our implementation, we used the Short Messaging Service (SMS) over the cellular network for the key exchange process. The Universal Mobile Telecommunication System (UMTS), an emerging standard for third generation (3G) wireless communications, has adopted an enhanced authentication and key agreement protocol [1]. The enhanced security of the 3G cellular systems enforces access control of users and mobile stations, data confidentiality, data integrity, and user identity privacy [5, 9, 10].

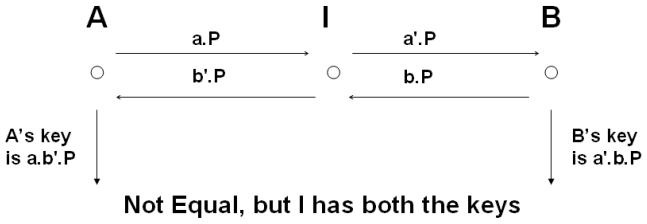


Figure 4. Man-In-The-Middle Attack during Key Agreement

Hence, in practice, it is sufficiently hard for an intruder to impersonate anybody else in the cellular network and tamper with messages. Ian will neither be able to impersonate himself as Alice or Bob, nor tamper with the messages exchanged during the key agreement protocol.

4.5 Loss of Device

In the event of the loss/theft of a device, the malicious user can connect to all the devices that are present in that device's address book. Though *SPACE* does not do anything explicitly to deal with this vulnerability, it assumes the existence of an external authority that deals with the general problems arising out of loss of a device and relies on the same for the security of our protocol. In practice, when a user loses his phone (and his SIM card), he notifies the phone manufacturer and the cellular provider who in turn have measures to track usage of the device and block the SIM card respectively.

5 Privacy Analysis

This section deals with the privacy implications of *SPACE*.

5.1 Hash Value based Identity

If the hash value broadcast in the Scan Phase was constant, an intruder can use it as the identity of a device to map all the places that the device was present e.g. to find if someone was in a particular location at different points in time; or if any one from a particular location moved to another location in the last ten minutes. So, if an intruder knows the hash of the contact detail of a device, he can easily locate all the places that the device visited by matching the hash value, thereby violating the privacy of the device even though he cannot invert the hash to obtain the contact detail. In our protocol, since we have a dynamic key for the hash function, the hash values are different for every broadcast.

5.2 Is any given contact detail present?

Assume that the adversary Ian wants to find if Bob is in Alice's address book. Ian can send a message pretending to be Bob. More generally, Ian can attempt to check the presence of a set of addresses in Alices address book.

In our protocol Alice responds with an encrypted nonce for the situation when she has Bob's address in her address book and does not send anything if the address is not present. This leaks the information Ian is seeking which causes a privacy concern. To address this we modify the protocol as follows. Now Alice sends a response in both situations:when she does not have the address she can encrypt with a random key. Ian will not be able to distinguish between the two responses. As before Alice will send an SMS if Bob's address was present but no key agreement was done. But if Ian is able to scan the atmosphere for SMS activity and analyze their pattern, he may be able to discover that Bob is indeed in the address book. We assume that Ian cannot scan the cellular network to isolate the needed cellular activity (SMS) from other normal activity when Alice exchanges messages corresponding to the key agreement protocol. Note that our extension costs an extra encryption and decryption step.

5.2.1 Are my contact details present?

A special case of the earlier attack is when Ian tries to determine the presence of his own address in Alices address book. An example is that authorities have a suspected mobile phone on hand and intend to find the set of people in a mall who have this number. In general, a clear policy has to be in place as to how to handle all the cases. Note that if Alice is in the mall, and Ian knows her number from the phone at hand, we expect Ian to detect her presence and make progress in setting up a connection.

Depending on whether Alice has Ian's address in her address book or not, Ian will receive the message corresponding to the key agreement protocol in the Authentication Phase. If this is a privacy concern for Alice, we can modify the Authentication phase in the protocol at an additional cost to Alice to send a challenge encrypted with her own address (this assumes address space is large enough, or is augmented to be large enough) before initiating the key agreement protocol. Alice initiates the key agreement protocol and the subsequent nonce challenge-response only if Ian responds correctly to the previously sent challenge. Otherwise she sends a nonce-challenge with a random key to Ian. Ian would not be able to respond to this challenge correctly and the protocol halts. But Ian will not be able to distinguish this random

challenge string from a valid challenge string and hence will not be able to make any conclusion about the presence of his address in Alice's address book.

If Alice did not have Ians contact detail he will not get any message corresponding to the key agreement protocol. But Ian cannot conclude from this because it could be that, either Alice does not have Ians contact detail or Alice already has an entry for Ian in his key table. So Ian cannot determine the presence of his contact detail in Alices address book.

6 IMPLEMENTATION

We implemented *SPACE* on smartphones running Windows Mobile 2003 SE operating system and communicate over Bluetooth (WLAN). The addresses we use are phone numbers in the cellular network. Alice hashes her phone number using SHA_1 and sends it to Bob. Bob verifies if the hash of any of the phone numbers in his phone book matches with the value sent by Alice. He subsequently sends over the hash of his number to Alice who performs the same check. In the first connection instance, Alice and Bob establish a shared secret key between them using the Elliptic Curve Diffie Hellman key agreement protocol [6] via SMS messages. The agreed key is stored for the purpose of authentication. The variables a , b and P used in the key agreement protocol were of sizes 160 bits each. Now, Alice and Bob can present a nonce to each other using this key and authenticate each other. We use the AES encryption algorithm for this purpose. The size of the nonces were 64 bits.

7 RELATED WORK

Bluetooth and IEEE 802.11x are the popular wireless local network standards. There has been considerable analysis on their security.

In Bluetooth, pairing is facilitated by the *initialization* key. This key is computed by a pair of devices using the Bluetooth addresses of each device, a random number, and a shared secret (PIN). The pairing session results in the *link* key that is unique for a pair of users and used for future communications. The security of the pairing process is dependent on the secrecy of the PIN. It is simple to crack the PIN if the communications occurring while the devices are paired is recorded [8]. Shaked et. al. [12] demonstrate that it is possible to crack a 4-digit PIN in 0.06 seconds using standard hardware. This makes it vulnerable to impersonation attacks. In contrast, SPACE does not suffer from these vulnerabilities and is specifically resistant to impersonation attacks.

In 802.11b authentication is performed by a challenge response procedure using a shared secret. After requesting authentication, the authenticator sends the initiator a 128-octet random number challenge. The initiator encrypts the challenge using the shared secret and transmits it back to the authenticator. A simple and powerful attack on this authentication mechanism is presented by Arbaugh et. al. [11]. First the intruder determines the pseudorandom string by recording the challenge (plaintext) and the response (ciphertext) and XOR-ing them. He then impersonates the victim by using the pseudorandom string to compute the response to subsequent challenges. This vulnerability towards plaintext/ciphertext attacks do not exist with SPACE as we do not send the unencrypted text at any stage.

8 CONCLUSIONS AND FUTURE WORK

We have proposed a novel protocol, *SPACE* for ad hoc connection establishment between mobile devices based on real-life relationships. We explored relevant security and privacy concerns with our protocol and augmented our protocol accordingly.

We understand that the address book entries are not the most secure basis for trust. There might be scenarios when we have contact details of people whom we do not entirely trust or not interested in sharing resources. We want to develop a mechanism to identify the preferred/trusted users in a contact list. One way is for users to manually mark the contact entries with whom they would prefer sharing resources. We can automate this by extracting information from the call-log/e-mails and obtain the set of contact entries with which a user has a high frequency of communication.

As a follow-up to Section 3 and 4, we aim to do a rigorous security and privacy analysis so that the protocol could be used for critical applications. We intend to incorporate a digital signature mechanism into the key agreement protocol and make it independent of the security policies of the underlying network. We also want to develop and test other useful applications on top of our protocol.

REFERENCES

[1] 3rd Generation Partnership Project; Technical Specification Group SA. Security Architecture, version 4.2.0, Release 4. In *3GPP*, 2001.

[2] A. A. Pirzada and C. McDonald. Establishing trust in pure ad-hoc networks. In *27th Australasian Computer Science Conference (ACSC) 26(1)*, pages 47–54, Jan 2004.

[3] Alfred J. Menezes , Scott A. Vanstone , Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press Inc, Boca Raton, Florida, 1996.

[4] E. Thomas and G. Xydis. Security Comparison: Bluetooth Communications vs. 802.11.

[5] G. Koien. An introduction to access security in UMTS. In *IEEE Wireless Communication, vol. 11, no. 1, pp. 818*, Jun 2004.

[6] I. F. Blake, G. Seroussi and N. P. Smart. Elliptic curves in cryptography. Technical report, London Math. Soc. Lecture Note Series, 2000.

[7] Lauter K. The Advantages Of Elliptic Curve Cryptography for Wireless Security. In *IEEE Wireless Communications*, Feb 2004.

[8] M. Jakobsson and S. Wetzel. Security Weaknesses in Bluetooth. In *RSA Security Conference - Cryptographer's Track, LNCS 2020*, 2001.

[9] M. Shin et. al. Wireless Network Security and Interworking. In *IEEE, vol. 94, pp 455-466*, Feb. 2006.

[10] Third Generation Partnership Project. 3GPP Technical Specifications, 3G security; Security Architecture (Release 6). 2003.

[11] W. Arbaugh et. al. Your 802.11 Wireless Network has No Clothes. In *IEEE Wireless Communications*, December 2002.

[12] Yaniv Shaked and Avishai Wool. Cracking the Bluetooth PIN. In *3rd international conference on Mobile systems, applications, and services (MOBISYS '05)*, 2005.

[13] Zhaoyu Liu, Anthony W. Joy, Robert A. Thompson. A Dynamic Trust Model for Mobile Ad Hoc Networks. In *10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS'04)*, pages 80–85, May 2004.

Exploiting Social Networks for Internet Search

Alan Mislove[†‡] Krishna P. Gummadi[†] Peter Druschel[†]

[†]Max Planck Institute for Software Systems, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany.
[‡]Rice University, 6100 Main Street, MS-132, Houston, TX 77005, USA.

1 INTRODUCTION

Over the last decade, the World Wide Web and Web search engines have fundamentally transformed the way people find and share information. Recently, a new form of publishing and locating information, known as online social networking, has become very popular. While numerous studies have focussed on the hyperlinked structure of the Web and have exploited it for searching content, few studies, if any, have examined the information exchange in online social networks.

In the Web, explicit links called hyperlinks between content (typically pages) are the primary tool for structuring information. Hyperlinks are used by authors to embed a page in the Web of related information, by human users to manually browse the Web, and by search engines to crawl the Web to index content, as well as to rank or estimate the relevance of content for a search query.

In contrast to the Web, no explicit links exist between the content (typically photos, videos, and blog postings) stored in social networks. Instead, explicit links between users, who generate or publish the content, serve as the primary structuring tool. For example, in social networking sites like MySpace [15], Orkut [17], and Flickr [4], a link from user *A* to user *B* usually indicates that *A* finds the information published by *B* interesting or relevant, or *A* implicitly endorses *B*'s content due to an established social relationship. Such social links enable users to manually browse for information that is likely of interest to them, and could be used by search tools to index and locate information. In this paper, we seek to understand whether these social links can be exploited by search engines to provide better results.

This paper makes three contributions: First, we compare the mechanisms for content publication and location in the Web and online social networks. We argue that search techniques could benefit from integrating the different mechanisms used to find relevant content in the Web and social networks. Second, we present results from an experiment in social network-based Web search to support our contention. Third, we outline the research challenges and opportunities in leveraging social networks for future Internet search.

The remainder of this paper is organized as follows. In Section 2, we first contrast how content is exchanged in social networks and the Web, and then speculate on the potential of integrating the different search techniques used in these systems. We evaluate the potential of our integrated approach to search using a social network-based experiment in Section 3. We discuss the research challenges that need to be addressed in order to realize such an integrated search system in Section 4. We present related work in Section 5 and conclude in Section 6.

2 THE WEB VERSUS SOCIAL NETWORKS

In this section, we compare the Web and social networking systems, with respect to their mechanisms for *publishing* and *locating* content.[1] Publishing refers to the mechanism by which content creators make information available to other users; it includes the way users relate their content to other content found in the system. Locating refers to the mechanism by which users find information relevant to them; it includes the ways users browse or search the content in the system.

2.1 The Web

In the Web, the content typically consists of Web pages written in HTML.

Publishing: Users publish content by placing documents on a Web server. An author places hyperlinks into her page that refer to related pages. She may also ask other authors to include links to her page in their pages. Often, such links are placed deliberately to ensure the page is indexed and ranked highly by search engines.

Locating: Today, the predominant way of locating information on the Web is via a search engine. Modern Web search engines employ sophisticated information retrieval techniques and impressive systems engineering to achieve high-quality search results at massive scale.

The key idea behind search engines like Google is to exploit the hyperlink structure of the Web to determine both the corpus of information they index and the relevance of a Web page relative to a given query [18]. This

[1] We ignore the mechanisms for distributing content between users as they are similar in both the Web and many current online social networks. In both systems, the content is transferred using HTTP over TCP, and the users navigate the systems using their Web browser.

approach has proven highly effective, because the incident links to a page are strong indicators of the importance or relevance of the page's content in the eyes of other users.

However, hyperlink-based search has some well known limitations. First, while Web search is very effective for relatively static information, it may under-rate or miss recently published content. For a new page to be noticed and appropriately ranked by a search engine, (a) it must be discovered and indexed by the search engine, (b) hyperlinks to the new page must be included in subsequently published or edited pages, and (c) all such links must then be discovered by the search engine.

Second, as search engines determine the relevance of a page by its incident hyperlinks, their rating reflects the interests and biases of the Web community at large. For instance, a search for "Michael Jackson" yields mostly pages with information about the pop star. Computer scientists, however, may find the Web page of a professor with the same name more relevant. Refining the search to find that page is possible but can be tricky, particularly if one does not recall the professor's current affiliation or field of specialization.

Finally, the hyperlink structure influences whether a page is included in a search engine's index. Unlinked pages and non-publicly accessible pages are not indexed. Many other pages are not indexed because the search engine deems them insufficiently relevant, due to their location in the hyperlink structure. As a result, obscure, special-interest content is less likely to be accessible via Web search.

2.2 Social Networks

Online social networking Web sites have recently exploded in popularity. Sites offer services for finding friends like MySpace [15], Orkut [17], and Friendster [6], for sharing photos like Flickr [4], for sharing videos like YouTube [24] and Google Video [8], and for writing blogs like LiveJournal [12] and BlogSpot [3]. These sites are extremely popular with users: MySpace claims to have over 100 million users, while Flickr and Orkut boast 2.5 million and 13 million users, respectively. MySpace recently has been observed to receive more page hits than Google [16].

Examples of online social networking, though, have existed for much longer. For instance, the common practice of placing content on the Web and sending its URL to friends or colleagues is essentially an instance of social networking. Typically, the author has no intention of linking the content; thus, the content remains invisible to users other than the explicit recipients of the URL. The content is advertised not via hyperlinks, but via links between users.

Publishing: Users publish content by posting it on a social networking site. Content is associated with the user who introduced it, and with users who explicitly recommend the content. Explicit links do not generally exist between content instances, and the content can be of any type. Often, the content is temporal in nature (e.g. blog postings), non-textual (e.g. photos and video clips), and may be of interest only to a small audience.

Independent of the content, users maintain links to other users, which indicate trust or shared interest. Links can be directed (indicating that the source trusts or is interested in the content of the target) or undirected (indicating mutual trust or interest in each other's content). Some systems maintain groups of users associated with different topics or interests; users can then join groups rather than specifying links to individual users. In some systems, the full social network graph is public; in others, only immediate neighbors of a node can view that node's other neighbors.

Locating: The predominant method of finding information in online social networks is to navigate through the social network, browsing content introduced or recommended by other users. Some sites also provide keyword-based search for textual or tagged content. Additionally, other sites have 'top-10' lists showing the most popular content, where the popularity is determined according to how often users have accessed the content or based on explicit recommendations provided by users.

Moreover, social networks enable users to find timely, relevant and reliable information. This is because users can browse adjacent regions of their social network, which likely consist of users with shared interests or mutual trust. Since the content can be non-textual, obscure, or short-lived, it may be hard to find by the way of Web search. For example, blog posts are generally of short-term interest, videos and photos are non-textual, and all three types of content tend to be of interest to a limited audience.

Content in social networks can also be rated rapidly, based on implicit and explicit feedback of a large community of content consumers. In contrast, Web search relies on the slower process of discovering hyperlinks in the Web, which are created by a relatively smaller number of content authors. Since content rating in social networks is performed by the content consumers, rather than the producers, content introduced into the network can by rated almost immediately.

2.3 Integrating Web search and social networks

Today, the information stored in different social networks and in the Web is mostly disjoint. Each system has its own method of searching information. While search companies have started to address this issue with specialized search tools for RSS-based news feeds and for

blogs, there is no unified search tool that locates information across different systems. Social network-based search methods are not generally used in the Web, though services like Google Scholar support search facilities tailored to a specific community. Given that end users access both the Web and the social networks from the same browsers, it seems natural to unify the methods to find information as well.

In this paper, we explore the idea of integrating Web search with search in social networks. We believe that such an approach could combine the strengths of both types of systems: simultaneously exploiting the information contained in hyperlinks, and information from implicit and explicit user feedback; leveraging the huge investment in conventional Web search, while also ranking search results relative to the interests of a social network; and locating timely, short-lived, non-textual or special-interest information alongside the vast amounts of long-lived and textual information on the Web.

3 PEERSPECTIVE: SOCIAL NETWORK-BASED WEB SEARCH

Our discussion above suggests that (a) a growing body of Internet content cannot be retrieved by traditional Web search as it is not well-connected to the hyperlinked Web, and that (b) social network links can be leveraged to improve the quality of search results. We are currently exploring the benefits of social networks-based Web search as part of the *PeerSpective* project. In this section, we describe a simple experiment we conducted to validate and quantify our two separate hypotheses.

3.1 Experimental methodology

We recruited a group of ten graduate students and researchers in our institute to share all Web content downloaded or viewed with one another. Each user runs a lightweight HTTP proxy, which transparently indexes all visited URLs. When a Google search is performed, the proxy transparently forwards the query to both Google as well as the peer proxies of other users in the social network. Each proxy executes the query on the local index and returns the result to the sender. The results are then collated and presented alongside the Google results as shown in Figure 1.

Our experimental PeerSpective prototype relies on the Lucene [13] text search engine and the FreePastry [5] peer-to-peer overlay. We configured Lucene to follow Google's query language. Also, we ranked the results obtained from PeerSpective by multiplying the Lucene score of a search result by the Google PageRank of that result and adding the scores from all users who previously viewed the result. Thus, PeerSpective's ranking takes advantage of both the hyperlinks of the Web (via Google's PageRank) and the social links of the user community.

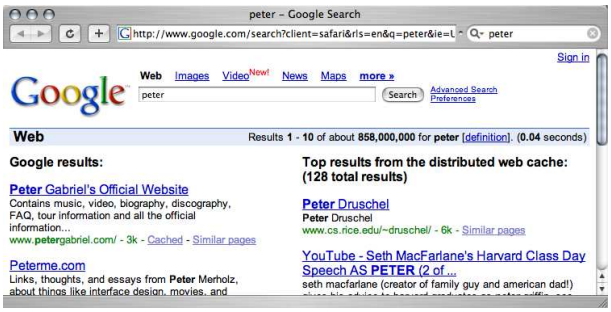


Figure 1: **Screenshot of our PeerSpective search interface**. Results from the distributed cache appear alongside the normal Google results.

We present measurements and experiences from a one month long experimental deployment. During this time, the 10 users issued 439,384 HTTP requests covering 198,492 distinct URLs. Only 25.9% of the HTTP requests were of content type `text/html` or `application/pdf`, meaning they could be indexed by our proxy. The remaining requests consisted of images, javascript, and other miscellaneous types.

Given that our user base is small, includes the authors, and represents a single community with highly specialized interests, we cannot claim that our results would be representative of a deployment with a larger, diverse user base. However, we believe our results indicate the potential of social network-based Web search. A more comprehensive study, which also considers Web access traces collected at the gateway router of a major university, is currently in progress.

3.2 Limits of hyperlink-based search

Even the best Web search engines do not index content that is not well linked to the general Web or content that is not publicly available. So, our first goal is to understand and quantify the Internet content that is viewed by users, but is not captured by the search engines. We would also like to know how much of this content is already indexed by another user in PeerSpective.

To estimate the limits of hyperlink-based search, we check what fraction of the URLs actually visited by the users are not indexed by Google. There are a number of reasons why a page may not be indexed by Google: (a) the page could be *too new*, such a as blog posting or news article; (b) the page could be in the *deep web* and not well-connected enough for Google to choose to crawl it; or (c) the page could be in the *dark web*, where it is not publicly available or is not referred to by any other page.

For each HTTP request, we checked whether Google's index contains the URL, and if some peer in PeerSpective has previously viewed the URL. Since search engines

URL	Too new	Deep web	Dark web
http://jwz.livejournal.com/413222.html	✓	✓	
http://www.mpi-sws.mpg.de/∼pkouznet/ ... /pres0031.html		✓	
http://sandiego.craigslist.org/w4m/179184549.html	✓	✓	
http://edition.cnn.com/2006/ ... /italy.nesta/index.html	✓		
http://72. ... .163/status.asp			✓
http://www.itv.com/news/ ... a8e4b6ea.html	✓		
http://www.stat.rice.edu/∼riedi/ ... /target21.html		✓	
http://amarok.kde.org/forum/index.php/board,9.20.html	✓	✓	

Figure 2: **Sample URLs that were not indexed by Google**. We manually inspected the URLS to determine the likely reason for not being in Google's index, as discussed in Section 3.2.

only index static HTML content, we considered only URLs of indexable content types which did not have any GET or POST parameters and ended in either `.html` or `.htm`. Further, we discarded URLs with an auto-refresh feature (such as the scoreboard sites for sports), as they would artificially bias the results against Google. This left us with 6,679 requests for 3,987 URLs.

Our analysis shows that Google's index covers only 62.5% of the requests, representing 68.1% of the distinct URLs. This implies that about one third of all URLs requested by our users cannot be retrieved by searching Google! Our analysis also showed that the union of the PeerSpective peer indexes covers about 30.4% of the requested URLs. While PeerSpective achieves only half of the coverage of Google's index, it does this with a much smaller size: at the end of the experiment, the PeerSpective indexes contained 51,410 URLs, compared to Google's index of over 8 billion URLs.

Additionally, we found that 13.3% of the URLs viewed were contained in PeerSpective but not in Google's index. These documents were not available via Google's search engine but had been requested before by someone in the peer network. This increase in coverage amounts to a 19.5% improvement by PeerSpective compared to normal Google search. It is worth noting that, for our small social network of computer science researchers, this improvement in coverage was possible by adding just a few thousand URLs to a Google index containing billions or URLs.

Our results naturally raise the question, what are these documents that are of a of interest to our users, but are not indexed by Google? We manually analyzed a number of such URLs and show a random sample of them in Figure 2. We additionally list the likely reasons why each URL does not appear in Google's index.

3.3 Benefits of social network-based search

Another challenge facing search engines is ranking all the indexed documents in the order of their relevance to a user's query. Ranking is crucial for search, as most users rarely go beyond the first few query results [20]. Our goal here is to study how often users click on query results from PeerSpective as opposed to Google. As shown in Figure 1, our users are presented with results from both Google and PeerSpective for every Google query.

During the course of the month, we observed 1,730 Google searches. While Google's first result page contained an average of 9.45 results, our smaller PeerSpective index resulted in an average of 5.17 results on the first page. Of the 1,730 queries, 1,079 (62.3%) resulted in clicks on one or more search result links, 307 (17.7%) were followed by a refined query, and after the remaining 344 (19.8%), the user gave up. We found that 933 (86.5%) of the clicked results were returned only by Google, 83 (7.7%) of the clicked results were returned only by PeerSpective, and 63 (5.7%) of the clicked results were returned by both. This amounts to a 9% improvement in search result clicks over Google alone, as 83 of the search result clicks would not have been possible without PeerSpective.

It should be kept in mind that this 9% improvement over Google, considered by many to be the gold standard for Web search engineering, was achieved by a simple, very small, social network-based system quickly put together by three systems researchers over a period of a few days. Based on our early experience, we feel that these results suggest inherent advantages of using social links for search, which could be exploited better with more careful engineering.

3.4 Discussion

To better understand the cases when PeerSpective search results outperform Google results, we manually analyzed the corresponding queries and result clicks. We show a random sample of the data we analyzed in Figure 3. We observed that the reasons for clicks on PeerSpective results fall into three categories:

Disambiguation: Some search terms have multiple meanings depending on the context. Search engines generally assume the most popular term definition. Social networks can take advantage of the fact that communities tend to share definitions or interpretation of such terms. An example for disambiguation is shown in Figure 3, where a user's query for "bus" yielded the local

Query	Page clicked on	Disambiguation	Ranking	Serendipity
bus	Saarbrücken bus schedule	✓	✓	
stefan	FIFA World Cup site			✓
peter	Peter Druschel's home page	✓		
serbian currency	XE.com exchange rates		✓	
coolstreaming	CoolStreaming INFOCOM paper		✓	
moose	Northwest Airlines' contract of carriage			✓
münchen	Peter Druschel's homepage			✓

Figure 3: **Sample search queries for which PeerSpective returned results not in Google**. The results are categorized into different scenarios discussed in Section 3.4.

bus schedule, as it is the page with this keyword that is most visited by local users in the network.

Ranking: Search engines rank all relevant documents and return the top of the resulting list. Social networks can inform and bias the ranking algorithm, since nearby users in the network often find similar sets of pages relevant. An example we observed is a search with the term "coolstreaming". A Google search ranks most highly popular sites (such as Wikipedia) discussing the CoolStreaming technique for P2P streaming of multimedia content. PeerSpective ranked the INFOCOM paper describing CoolStreaming at the top, as it is most relevant to our researchers.

Serendipity: While browsing the Web, users often discover interesting information by accident, clicking on links that they had not intended to query for. This process, termed serendipity, is an integral part of the Web browsing experience. Search results from PeerSpective provide ample opportunity for such discoveries. For example, while looking for information about "München" (Munich), one of our users discovered that a fellow researcher attended school in München, thus finding a convenient source of information about the city.

4 OPPORTUNITIES AND CHALLENGES

Online social networking enables new forms of information exchange in the Internet. First, end users can very easily and conveniently publish information, without necessarily linking it to the wider Web. Second, social networks make it possible to locate and access information that was previously exchanged by "word of mouth", that is, by explicit communication between individuals. Third, unlike Web search engines, which organize the world of information according to popular opinion, social networks can organize the world of information according to the tastes and preferences of smaller groups of individuals.

We see great potential in the integration of the Web and social network search technologies. Such an integration can provide unified access to a significantly larger body of online information than what is currently available in the shallow Web. We presented evidence that the integration can also improve the quality of Web search results by ranking the results relative to the interests and biases of groups of individuals. In this section, we discuss research opportunities and challenges associated with realizing this vision.

Privacy: Participants in a social network must be willing to disclose which information they find interesting and relevant. This creates a tension between the privacy concerns of individuals and the effectiveness of the social network, which depends on the willingness of individuals to share information. In small social networks of mutually trusting participants (e.g., family members or close friends) the problem reduces to access control. However, in larger social networks (e.g., all researchers in computer networking), a solution that is acceptable to users would require mechanisms to control information flow and anonymity.

Membership and clustering of social networks: In general, an individual may be a participant in multiple social networks (e.g., networks related to professional interests, networks related to hobbies, and networks related to family and friends). This raises many questions. Are there automated mechanisms by which we can infer the social links between users? For instance, by observing email exchange between users, or by considering similarity in content browsed or stored between pairs of users. Similarly, can we automatically identify different clusters of communities associated with certain interests? In the absence of such techniques, users have to explicitly declare and manage their social network memberships. Finally, if a user participates in many social clusters, how should a search query be resolved with respect to the different clusters?

Content rating and ranking: The use of social network-based search techniques enables new approaches to ranking search results. There are many alternatives that could be explored: should we use global page rank, as in Google, or should we use a local page rank specific to the social network? Should content be ranked based on the number of users who have viewed or stored the content, or should the ranking be based on explicit user ratings of the content? Furthermore, how should the search results from the social network be displayed or ranked relative to the Google results?

System architecture: Should the system be centralized or distributed? A centralized architecture, similar to current Web search engines, may raise concerns about privacy, trust and market dominance. Also, a centralized approach may not scale with the bandwidth requirements of a central data store or the number of different social networks. A decentralized architecture, on the other hand, faces challenges of its own. Building even a conventional Web search engine in a decentralized fashion is an open research problem. Adding decentralized social network search requires scalable, index-based search algorithms, and appropriate mechanisms to ensure privacy.

5 RELATED WORK

Several projects have looked at replacing the functionality of the large centralized Web search engines with a decentralized system, built from contributing users' desktops [11]. Both Minerva [2] and YaCy [22] implement a peer-to-peer Web search engine without any points of centralization. Additionally, other projects [10, 19] have examined replacing the centralized PageRank computation of Google with a decentralized approach. All of these projects, though, are primarily focused on replacing the functionality of existing centralized search engines with a decentralized architecture.

A few systems have looked at query personalization, or taking a user's preferences and interests into account when ranking pages. Most notably, A9 [1] and Google Personalized Search [7] allow users to create profiles to which search results are tailored. There has also been much research into methods for accurately personalizing search queries [9, 21]. While these projects are concerned with personalization, our work is complementary and examines the ability to use social links to improve search results.

Lastly, a number of projects have looked at using social networks to aid a variety of applications. Notable distributed systems projects include SPROUT [14], which uses the trust of social links to increase the probability of successful DHT routing, and Maze [23], which allows users to create friends in the file sharing network.

6 CONCLUSION

In this paper, we examined the potential for using online social networks to enhance Internet search. We analyzed the differences between the Web and social networking systems in terms of the mechanisms they use to publish and locate useful information. We discussed the benefits of integrating the mechanisms for finding useful content in both the Web and social networks. Our initial results from a social networking experiment suggest that such an integration has the potential to improve the quality of Web search experience. Finally, we outlined research challenges in leveraging online social networks to build search systems for the future Internet.

REFERENCES

[1] A9 Search. `http://www.a9.com`.

[2] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. MINERVA: Collaborative P2P search. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB'05)*, August 2005.

[3] BlogSpot. `http://www.blogspot.com`.

[4] Flickr. `http://www.flickr.com`.

[5] FreePastry Project. `http://www.freepastry.org`.

[6] Friendster. `http://www.friendster.com`.

[7] Google Personalized Search. `http://www.google.com/psearch`.

[8] Google Video. `http://video.google.com`.

[9] G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the 12th International World Wide Web Conference (WWW'03)*, May 2003.

[10] K.Sankaralingam, S. Sethumadhavan, and J.C.Browne. Distributed PageRank for P2P systems. In *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12)*, June 2003.

[11] J. Li, B. T. Loo, J. Hellerstein, F. Kaashoek, D. R. Karger, and R. Morris. On the feasibility of peer-to-peer web indexing and search. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, February 2003.

[12] LiveJournal. `http://www.livejournal.com`.

[13] Lucene Search Engine. `http://lucene.apache.org`.

[14] S. Marti, P. Ganesan, and H. Garcia-Molina. DHT routing using social links. In *Proceedings of the 3rd International Workshop on Peer-to-Peer Systems (IPTPS'04)*, February 2004.

[15] MySpace. `http://www.myspace.com`.

[16] MySpace is the number one website in the U.S. according to hitwise. HitWise Press Release, July, 11, 2006. `http://www.hitwise.com/press-center/hitwiseHS2004/social-networking-june-2006.php`.

[17] Orkut. `http://www.orkut.com`.

[18] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[19] J. X. Parreira, D. Donato, S. Michel, and G. Weikum. Efficient and decentralized PageRank approximation in a peer-to-peer web search network. In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB'06)*, September 2006.

[20] B. Smyth, E. Balfe, O. Boydell, K. Bradley, P. Briggs, M. Coyle, and J. Freyne. A live-user evaluation of collaborative web search. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, July 2005.

[21] J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *Proceedings of the 28th Annual International ACM SIGIR Conference (SIGIR'05)*, August 2005.

[22] YaCy Search Engine. `http://www.yacy.net`.

[23] M. Yang, H. Chen, B. Y. Zhao, Y. Dai, and Z. Zhang. Deployment of a large-scale peer-to-peer social network. In *Proceedings of the 1st Workshop on Real, Large Distributed Systems (WORLDS'04)*, December 2004.

[24] YouTube. `http://www.youtube.com`.

Free Riding in BitTorrent is Cheap

Thomas Locher[1], Patrick Moor[2], Stefan Schmid[1], Roger Wattenhofer[1]
[1] Computer Engineering and Networks Laboratory (TIK), ETH Zurich, 8092 Zurich, Switzerland
{lochert, schmiste, wattenhofer}@tik.ee.ethz.ch
[2] Google Inc., Mountain View, CA 94043, USA
pmoor@google.com

ABSTRACT

While it is well-known that BitTorrent is vulnerable to selfish behavior, this paper demonstrates that even entire files can be downloaded without reciprocating at all in BitTorrent. To this end, we present *BitThief*, a free riding client that never contributes any real data. First, we show that simple tricks suffice in order to achieve high download rates, *even in the absence of seeders*. We also illustrate how peers in a swarm react to various sophisticated attacks. Moreover, our analysis reveals that *sharing communities*—communities originally intended to offer downloads of good quality and to promote cooperation among peers—provide many incentives to cheat.

1 INTRODUCTION

As pure peer-to-peer (p2p) systems are completely decentralized and resources are shared directly between participating peers, all p2p systems potentially suffer from *free riders*, i.e. peers that eagerly consume resources without reciprocating in any way. Not only do free riders diminish the quality of service for other peers, but they also threaten the existence of the entire system.

For that reason, it is crucial for any system without centralized control to incorporate a rigorous incentive mechanism that renders freeloading evidently unattractive to selfish peers. Unfortunately, however, many solutions so far either could easily be fooled or were unrealistically complex. Bram Cohen's BitTorrent protocol heralded a paradigm shift as it demonstrated that cooperation can be fostered among peers interested in the same file, and that concentrating on one file is often enough in practice. The fair sharing mechanism of BitTorrent is widely believed to strongly discourage freeloading behavior.

Contrary to such belief, we show that BitTorrent in fact does not provide sufficient incentives to rule out free riding. The large degree of cooperation observed in BitTorrent swarms is mainly due to the widespread use of obedient clients which willingly serve all requests from other peers. We have developed our own BitTorrent client *BitThief*[1] that never serves any content to other peers. With the aid of this client, we demonstrate that a peer can download content fast *without uploading any data*. Surprisingly, BitThief always achieves a high download rate, and in some experiments has even outperformed the *official client*. Moreover, while *seeders* ("altruistic peers") clearly offer the opportunity to freeload, we are even able to download content quickly if we ignore seeders and download solely from other peers that do not possess all pieces of the desired content (*leechers*). This implies that the basic piece exchange mechanism does not effectively restrain peers from freeloading.

Sharing communities are also investigated in this paper. By banning users with constantly low sharing ratios or by denying them access to the newest torrents available, such communities encourage users to upload more than they download, i.e., to keep their sharing ratio above 1. We will show that sharing communities are particularly appealing for free riders, and that cheating is easy.

We believe that the possibility to freeload which does not come at the cost of a considerably reduced quality of service (e.g., download rate) is attractive for users: Not only because wasting more expensive upload bandwidth is avoided, but also because—in contrast to downloading—merely the *distribution* of copyrighted media content such as music or video shared in p2p networks is unlawful in certain countries.

However, as more and more users decide to free ride, the usefulness of a p2p system will naturally decline. Thus, spreading such freeloading clients might prove to be an effective attack for corporations fighting the uncontrolled distribution of their copyrighted material.

2 BITTORRENT

The main mechanisms applied by *BitTorrent* are described in [4]; for additional resources including a detailed technical protocol, the reader is referred to *www.bittorrent.org*. Basically, BitTorrent is a p2p application for sharing files or collections of files. In order to participate in a *torrent download*, a peer has to obtain a *torrent metafile* which contains information about the content of the torrent, e.g. file names, size, tracker addresses, etc. A tracker is a centralized entity that keeps track of all the peers (TCP endpoints) that are downloading in a specific torrent swarm.[2] Peers obtain contact information of other participating peers by announcing themselves to the tracker on a regular basis. The data to be shared is divided into pieces whose size is specified in the metafile (usually a couple of thousand pieces per torrent). A hash of each piece is also stored in the metafile, so that the downloaded data can be verified piece

[1] Available at http://dcg.ethz.ch/projects/bitthief/.

[2] Recently, a distributed tracker protocol has been proposed. It is implemented by most modern clients.

by piece. Peers participating in a torrent download are subdivided into *seeders* which have already downloaded the whole file and which (altruistically) provide other peers with any piece they request, and *leechers* which are still in progress of downloading the torrent. While seeders upload to all peers (in a round robin fashion), leechers upload only to those peers from which they also get some pieces in return. The peer selection for uploading is done by unchoking a fixed number of peers every ten seconds and thus enabling them to send requests. If a peer does not contribute for a while it is choked again and another peer is unchoked instead.

The purpose of this mechanism is to enforce contributions of all peers. However, each leecher periodically *unchokes* a neighboring leecher, transferring some data to this neighboring peer for free (called *optimistic unchoking* in BitTorrent lingo). This is done in order to allow newly joined peers without any pieces of the torrent to *bootstrap*. Clearly, this unchoking mechanism is one weakness that can be exploited by BitThief.

3 BitThief: A Free Riding Client

In this section we provide evidence that, with some simple tricks, uploading can be avoided in BitTorrent while maintaining a high download rate. In particular, our own client *BitThief* is described and evaluated. BitThief is written in Java and is based on the official implementation[3] (written in Python, also referred to as *official client* or *mainline client*), and the *Azureus*[4] implementation. We kept the implementation as simple as possible and added a lot of instrumentational code to analyze our client's performance. BitThief does not perform any chokes or unchokes of remote peers, and it never announces any pieces. In other words, a remote peer always assumes that it interacts with a newly arrived peer that has just started downloading. Compared to the official client, BitThief is more aggressive during the startup period, as it re-announces itself to the tracker in order to get many remote peer addresses as quickly as possible. The tracker typically responds with 50 peer addresses per announcement. This parameter can be increased to at most 200 in the announce request, but most trackers will trim the list to a limit of 50. Tracker announcements are repeated at an interval received in the first announce response, usually in the order of once every 1800 seconds. Our client ignores this number and queries the tracker more frequently, starting with a configurable interval and then exponentially backing off to once every half an hour. Interestingly, during all our tests, our client was not banned by any of the trackers and could thus gather a lot of peers. The effect of our aggressive behavior is depicted in Figure 1. Finally, note that it would also be possible to make use of the *distributed tracker protocol*.[5] This protocol is useful if the main tracker is not operational. Thus far, we have not incorporated this functionality into our client however.

[3] See http://bittorrent.com/.

[4] See http://azureus.sourceforge.net/.

[5] See http://www.bittorrent.org/Draft_DHT_protocol.html.

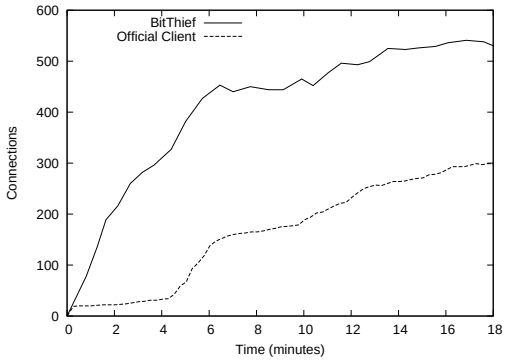


Figure 1: Number of open connections over time. In comparison to the official client, BitThief opens connections much faster.

Having a large number of open connections improves the download rate twofold: First, connecting to more seeders allows our client to benefit more often from their round robin unchoking periods. Second, there will be more leechers in our neighborhood that include BitThief in their periodical optimistic unchoke slot. Opening more connections increases download speed linearly, as remote peers act independently of the number of our open connections. However, note that opening two connections to the same peer does not help, as the official client, Azureus, and presumably all other clients as well immediately close a second connection originating from the same IP address.

Our experiments with BitThief demonstrate that the common belief that the performance will degrade if a large number of TCP connections is maintained simultaneously is unfounded. On the contrary, more connections always help to increase the download rate when using BitThief. The reason why the total number of TCP connections is kept small in BitTorrent might be that a moderate number of connections suffice to saturate the average user's bandwidth when following the real protocol, and no further gain could be achieved by connecting to more peers.

In contrast to other BitTorrent clients, BitThief does not apply the so-called *rarest-first policy*, but uses a simpler piece selection algorithm instead: We fetch whatever we can get. If our client is unchoked by a remote peer, it picks a random missing piece. Our algorithm ensures that we *never* leave an unchoke period unused. Furthermore, just like all other BitTorrent clients, we strive to complete the pieces we downloaded partially as soon as possible in order to check them against the hash from the metafile and write them to the harddisk immediately.

3.1 Seeders

We first tested the client on several torrents obtained from *Mininova*[6] and compared it to the official client.[7] By default, the official client does not allow more than 80 connections. In order to ensure a fair comparison, we removed this limitation and permitted the client to open up to 500

[6] See http://www.mininova.org/.

[7] Official client vers. 4.20.2 (linux source). Obtained from bittorrent.com, used with parameters: `--min_peers 500 --max_initiate 500 --max_allow_in 500`.

	Size	Seeders	Leechers	μ	σ
A	170MB	10518 (303)	7301 (98)	13	4
B	175MB	923 (96)	257 (65)	14	8
C	175MB	709 (234)	283 (42)	19	8
D	349MB	465 (156)	189 (137)	25	6
E	551MB	880 (121)	884 (353)	47	17
F	31MB	N/A (29)	N/A (152)	52	13
G	798MB	195 (145)	432 (311)	88	5

Table 1: Characteristics of our test torrents. The numbers in parentheses represent the maximum number of connections BitThief maintained concurrently to the respective peer class and is usually significantly lower than the peer count the tracker provided. μ and σ are the average and standard deviation of the official client's download times in minutes. The tracker of Torrent F did not provide any peer count information. Based on the number of different IP addresses our client exchanged data with, we estimate the total number of peers in this torrent to be more than 340.

connections. In a first experiment, we did not impose any restrictions on our client, in particular, BitThief was also permitted to download from seeders. The tests were run on a PC with a public IP address and an open TCP port, so that remote peers could connect to our client. We further blocked all network traffic to or from our university network, as this could bias the measurements. The properties of the different torrents used in this experiment are depicted in Table 1. Note that the tracker information is not very accurate in general and its peer count should only be considered a hint on the actual number of peers in the torrent.

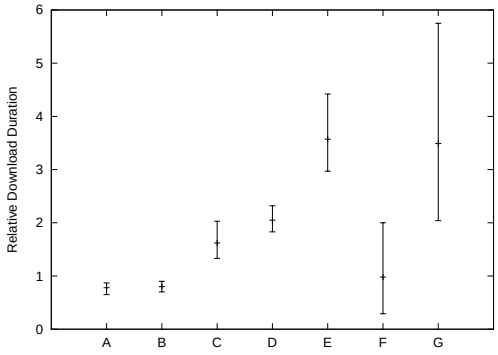


Figure 2: Relative download times for six torrents. The download time of the official client is normalized to 1.0. Every torrent was downloaded three times with both clients. The plot shows relative download times with the fastest run at the lower end of the bar, the average running time at the level of the horizontal tick mark, and the slowest run at the upper end of the bar.

The results are summarized in Figure 2. As a first observation, note that in every experiment, BitThief succeeded eventually to download the entire file. More interestingly, the time required to do so is often not much longer than with uploading! Exceptions are Torrents E and G, where there are relatively few seeders but plenty of leechers. In that case, it takes roughly four times longer with our client. However, the download came at a large cost for the official client as it had to upload over 3.5GB of data. Torrents A, B and F also offer valuable insights: In those torrents, BitThief was, on average, slightly faster than the official client, which uploaded 232MB in a run of torrent A and 129MB in a run of Torrent B. We conclude that in torrents with many peers, particularly seeders, and in torrents for small files, BitThief seems to have an advantage over the official client, probably due to the aggressive connection opening.

3.2 Leechers

In this section, we further constrain BitThief to only download from other leechers. Interestingly, as we will see, even in such a scenario, free riding is possible.

Seeders are identified by the *bitmask* the client gets when the connection to the remote peer is established, and the *have-message* received every time the remote peer has successfully acquired a new piece. As soon as the remote peer has accumulated all pieces, we immediately close the connection. We conducted the tests at the same time as in Section 3.1 and also used the same torrents. The running times are depicted in Figure 3. It does not come as a surprise that the average download time has increased. Nevertheless, we can again see that all downloads finished eventually. Moreover, note that the test is slightly unfair for BitThief, as the official client was allowed to download not only from the leechers, but also from all seeders! In fact, in some swarms only a relatively small fraction of all peers are leechers. For example in Torrent C, merely 15% are leechers, and BitThief can thus download from less than a sixth of all available peers; nevertheless, BitThief only requires roughly 5 times longer than the official client.

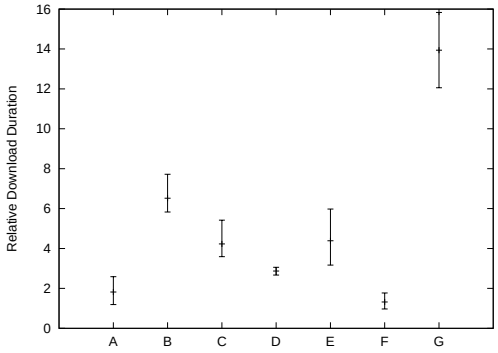


Figure 3: Relative download times of BitThief for six torrents *without downloading from any seeders*. The download time of the official client is normalized to 1.0. As in the first experiment, the torrents were downloaded three times with the official client and three times using BitThief restricted to download from leechers only. The bars again represent the same minimum, average and maximum running times.

We conclude that even without downloading from seeders, BitThief can download the whole torrent from leechers exclusively. Therefore, it is not only the seeders which provide opportunities to free ride, but the leechers can be exploited as well.

3.3 Further Experiments

The measurements presented so far have all been obtained through experiments on the Internet and hence were subject to various external effects. For example, in case BitThief

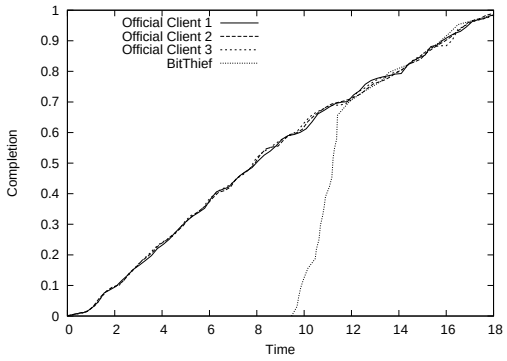


Figure 4: Download times for three official clients and one BitThief client in the presence of a slow seeder. BitThief starts downloading 9 minutes later than the other clients, but catches up quickly. Ultimately, all clients finish the download roughly at the same time.

was allowed to download from seeders, it sometimes downloaded at a high rate, but then—a few minutes later—the download rate declined abruptly due to a powerful seeder having left the network. In order to get reproducible results, we set up a pet network environment on a host, consisting of a private tracker, a configurable number of official clients as seeders and leechers, and one instance of our own client. We evaluated different scenarios. In the following, our main findings will be summarized briefly.

In scenarios with many seeders and only very few leechers, our client will download most data from seeders. As the leechers often do not fill up all their upload slots with other leechers,our client is unchoked all the time, yielding a constant download rate.

More interesting are scenarios with a small number of seeders. A fast seeder is able to push data into the swarm at a high rate and all the leechers can reciprocate by sharing the data quickly with their upstreams fully saturated. In this situation, it is difficult for our client to achieve a good downstream: We only get a small share of the seeders' upstream and all the other leechers are busy exchanging pieces between them. Hence, we only profit from the optimistic unchoke slots, which results in a poor performance. However, note that many leechers will turn into seeders relatively soon and therefore our download rate will increase steadily.

A slow seeder is not able to push data fast enough into the swarm, and the leechers reciprocate the newly arrived pieces much faster without filling all their upload slots. Although BitThief cannot profit from the seeders, it can make use of the leechers' free upload slots. The attainable download rate is similar to the one where there are many seeders. The download rate will go down only when BitThief has collected all pieces available in the swarm. When a new piece arrives, the leechers will quickly exchange it, enabling BitThief to download it as well with almost no delay. An experiment illustrating this behavior is given in Figure 4. Note that the execution shown in the figure is quite idealistic, as there are no other leechers joining the torrent over time.

In summary, the results obtained from experiments on the Internet have been confirmed in the experiments conducted in our pet network.

3.4 Exploiting Sharing Communities

Finding the right torrent metafile is not always an easy task. There exist many sites listing thousands of torrents (e.g., Mininova), but often the torrents' files are not the ones mentioned in the title or are of poor quality. Therefore, a lot of *sharing communities* have emerged around BitTorrent. These communities usually require registration on an invitation basis or with a limit on the number of active users. Finding good quality torrents in these communities is much more convenient than on public torrent repositories. Sharing communities usually encourage their users to upload at least as much data as they download, i.e., to keep their sharing ratio above 1. This is achieved by banning users with constantly low sharing ratios or by denying them access to the newest torrents available.

Andrade et al. [2] studied these communities and analyzed how sharing ratio enforcement influences seeding behavior. The authors find that seeders are staying in a torrent for longer periods of time, i.e., typically the majority of peers are seeders. These communities thus exhibit ideal conditions for BitThief, provided that we can find ways to access and stay in this communities without uploading.

We have found that this can often be done by simply pretending to upload. The community sites make use of the tracker announcements which every client performs regularly. In these announcements the client reports the current amount of data downloaded and uploaded. These numbers are stored in a database and used later on to calculate the sharing ratios. The tracker typically does not verify these numbers, although, in our opinion, it would be possible to expose mischievous peers: For instance, in a torrent with 100 seeders and just one leecher, it looks suspicious if the leecher is constantly announcing large amounts of uploaded data. Alternatively, the sum of all reported download and upload amounts could be analyzed over different torrents and time periods, in order to detect and ban dishonest peers.

The tracker can also be cheated easily: Clients can announce bogus information and fake peers so that the tracker's peer list fills up with dozens of clients which do not exist. The seeder and leecher counts reported by the tracker can therefore be misleading as there are usually not that many real peers downloading a given torrent. Even worse, peers asking a tracker for other peers can get a lot of invalid or stale information, which makes torrent starts slow.

An alternative is used by recent BitTorrent clients: A distributed tracker protocol which manages the torrent swarm. The technique of faking tracker announcements has been used in a couple of torrents in our tests and we now have a sharing ratio of 1.4 on *TorrentLeech*[8] without ever uploading a single bit.

An example which emphasizes how dramatic the difference between a community internal and an external

[8] See http://torrentleech.org/.

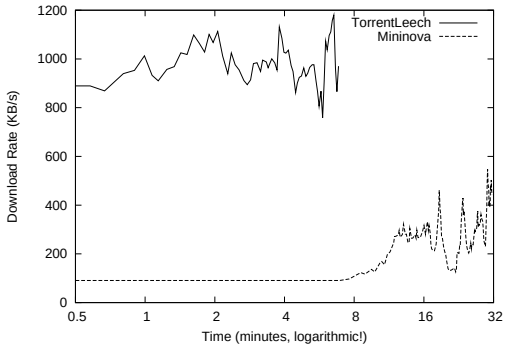


Figure 5: BitThief's download speed: comparison between a community version of a torrent and a torrent of the same file found on Mininova.

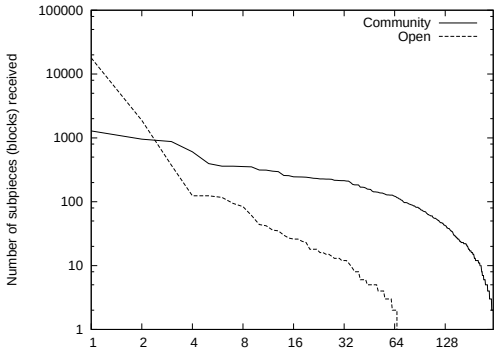


Figure 6: The logarithmically scaled list of peers ordered according to the number of provided blocks is plotted on the x-axis. The file size was 350 MB. On Mininova (*open*) and on TorrentLeech (*community*), BitThief connected to 309 and 349 peers, respectively. In the community network, the distribution is more balanced and BitThief is able to download from much more peers, while only a few peers contributed a large fraction of all blocks in the public torrent.

download can be, is given in Figure 5. We used a torrent that was published on TorrentLeech approximately 12 hours before conducting this experiment and looked for the same one on Mininova, where it had appeared 4 hours earlier. The torrent was 359MB in size on TorrentLeech and slightly smaller (350MB) on Mininova. We first downloaded the torrent three times from Mininova, then three times from TorrentLeech. The Mininova runs took 32/32/37 minutes, while on TorrentLeech the runs completed in 7:25/7:08/7:08 minutes, respectively. This is more than four times faster. Considering that there were only 25 (24 seeders, one leecher) peers in the TorrentLeech swarm and more than 834 (531 seeders, 303 leechers) peers in the other swarm, this is surprising.

As far as the individual contributions of the peers are concerned, we observed the following. While BitThief tends to benefit more from certain peers, generally seeders, in public torrents, a much larger fraction of all peers provides a considerable share of the file in sharing communities, and the distribution across peers is more balanced. This is probably due to the community peers' desire to boost their sharing ratios by uploading as much as possible. An experiment illustrating this point is depicted in Figure 6.

4 SOPHISTICATED ATTACKS

While simple tricks often yield a good performance, BitTorrent has proved to be quite robust against certain more sophisticated attacks.

First, we have investigated an exploit proposed in [10], which truly violates the BitTorrent protocol: The selfish client announces pieces as being available even if it does not possess them. If such an unavailable piece is requested by a remote peer, the client simply sends random data (garbage). As only the integrity of whole pieces can be checked, the remote peer cannot verify the subpiece's correctness. Note that this behavior cannot be considered free riding in the pure sense, but it is a strategy that does not require to upload any *valid* user data.

In a first implementation, all requests are answered by uploading entire garbage pieces. As has already been pointed out in [10], this approach is harmful: Both the official client and Azureus store information from whom they have received subpieces and will thus immediately ban our IP address once the hash verification fails.[9] Consequently, we have tried to answer all requests for a piece except for one subpiece, which would force the remote peer to get that subpiece from a different peer. The idea is that the remote peer cannot tell which peer uploaded the fake data, as it might as well be the other peer which only supplied one subpiece. While the official client can indeed be fooled this way, Azureus is smarter and uses an interesting approach: Once it has determined that the piece is not valid, it looks up from which peer it received most subpieces. The piece is then reserved for that peer, and Azureus aims at fetching all remaining subpieces from the same peer. When refusing to answer these requests, the connection stalls, and eventually our IP address is banned. We have tried several tricks to circumvent these problems, but came to the conclusion that uploading random garbage, in any way, does not improve performance.

When establishing connections, peers inform each other about their download status by sending a list of pieces that they have already successfully downloaded. While the connection is active, peers send messages to each other for each new piece they downloaded. Therefore, a peer always know the progress of its neighbors. We sought to measure the influence that this information has on a remote peer. Currently, BitThief sends an empty list of available pieces during connection setup and it does not inform the remote peer about any new pieces it acquires. We tried announcing different percentages of all the pieces at the beginning of the connection, but our experiments showed that the performance is independent of the percentage, as long as not all of the pieces are available. However, announcing 100% of the pieces has disastrous consequences, as the remote peer considers BitThief a seeder and therefore does not respond to any piece requests.

[9] Note that an appealing solution would be to fake entire pieces by using contents yielding the same hash values. Unfortunately, however, the computation of such SHA-1 hash collisions is expensive and would yield huge tables which cannot be stored in today's databases.

BitThief profits from the optimistic unchoke slots of leechers and from the round robin unchoke scheme of seeders. Thus, a client could possibly increase the chance of being unchoked by being present in the remote peer's neighborhood more than once. This is known as a *Sybil attack* [5]. However, this attack involves opening two or more connections to a remote peer. Both the official client and Azureus prevent such behavior. If multiple IP addresses are available, it would be an easy task to extend the client in a way to fake two entities and trick remote peers. The peers would gladly open a connection to both external addresses and thus our download rate might increase up to twofold.

5 Related Work

In 2000, Adar and Huberman [1] noticed the existence of a large fraction of free riders in the file sharing network *Gnutella*. The problem of selfish behavior in peer-to-peer systems has been a hot topic in p2p research ever since, e.g. [8, 12], and many mechanisms to encourage cooperation have been proposed, for example in [6, 7, 11, 13, 14].

BitTorrent [4] has incorporated a fairness mechanism from the beginning. Although this mechanism has similarities to the well known *tit-for-tat mechanism* [3], the mechanism employed in BitTorrent distinguishes itself from the classic tit-for-tat mechanism in many respects [9]. This fairness mechanism has also been the subject of active research recently. Based on PlanetLab tests, [9] has argued that BitTorrent lacks appropriate rewards and punishments and therefore peers might be tempted to freeload. The authors further propose a tit-for-tat-oriented mechanism based on the iterated prisoner's dilemma [3] in order to deter peers from freeloading. However, in their work, a peer is already considered a free rider if it contributes considerably less than other peers. We, on the other hand, aim at attaining fast downloads strictly without uploading any data. This is often desirable, since in many countries downloading certain media content is legal whereas uploading is not.

The paper closest to our work is by Liogkas et al. [10]. The authors implement three selfish BitTorrent exploits and evaluate their effectiveness. They come to the conclusion that while peers can sometimes benefit slightly from being selfish, BitTorrent is fairly robust. Our work extends [10] in that, rather than concentrating on individual attacks, we have implemented a client that combines several attacks (an open question in [10]). In contrast to our work, the authors examine the effect of free riders on the *overall system* and argue that the quality of service is not severely affected by the presence of some peers that contribute only marginally. We focus strictly on maximizing the download rate of a *single, selfish peer*, regardless of what effect this peer has on the system.

Finally, [2] has studied the cooperation in *BitTorrent communities*. It has been shown that community-specific policies can boost cooperation. In our work, we have demonstrated that cheating is often easy in communities and selfish behavior even more rewarding.

6 Outlook

In a first thread of future research, we aim at incorporating further selfish attacks such as *collusion* into BitThief. Moreover, current trends such as *ISP caching*[10] could also introduce new potential exploits.

In a second thread of research, we extend our BitThief client such that it truly enforces cooperation among peers. For this purpose, the *Fast Extension*[11] might serve as a promising starting point. A challenging problem which has to be addressed is to find a mechanism that applies some kind of tit-for-tat algorithm for older peers in the system, while at the same time efficiently solving the *bootstrap problem* of newly joining peers: As these new peers inherently do not have any data to share, they must be provided with some "venture capital".

References

[1] E. Adar and B. A. Huberman. Free Riding on Gnutella. *First Monday*, 5(10), 2000.

[2] N. Andrade, M. Mowbray, A. Lima, G. Wagner, and M. Ripeanu. Influences on Cooperation in BitTorrent Communities. In *Proc. 3rd ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems (P2PECON)*, 2005.

[3] R. Axelrod. The Evolution of Cooperation. *Science*, 211(4489):1390-6, 1981.

[4] B. Cohen. Incentives Build Robustness in BitTorrent. In *Proc. 1st Workshop on Economics of Peer-to-Peer Systems (P2PECON)*, 2003.

[5] J. R. Douceur. The Sybil Attack. In *1st International Workshop on Peer-to-Peer Systems (IPTPS), Cambridge, MA, USA*, pages 251–260, 2002.

[6] M. Feldman and J. Chuang. Overcoming Free-Riding Behavior in Peer-to-Peer Systems. *ACM Sigecom Exchanges*, 6, 2005.

[7] D. Grolimund, L. Meisser, S. Schmid, and R. Wattenhofer. Havelaar: A Robust and Efficient Reputation System for Active Peer-to-Peer Systems. In *1st Workshop on the Economics of Networked Systems (NetEcon), Ann Arbor, Michigan, USA*, June 2006.

[8] D. Hughes, G. Coulson, and J. Walkerdine. Free Riding on Gnutella Revisited: The Bell Tolls? *IEEE Distributed Systems Online*, 6(6), 2005.

[9] S. Jun and M. Ahamad. Incentives in BitTorrent Induce Free Riding. In *Proc. 3rd ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems (P2PECON)*, 2005.

[10] N. Liogkas, R. Nelson, E. Kohler, and L. Zhang. Exploiting BitTorrent For Fun (But Not Profit). In *Proc. 5th Itl. Workshop on Peer-to-Peer Systems (IPTPS)*, 2006.

[11] S. Sanghavi and B. Hajek. A New Mechanism for the Free-rider Problem. In *Proc. 3rd ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems (P2PECON)*, 2005.

[12] J. Shneidman and D. C. Parkes. Rationality and Self-Interest in Peer to Peer Networks. In *Proc. 2nd Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.

[13] K. Tamilmani, V. Pai, and A. Mohr. SWIFT: A System with Incentives for Trading. In *Proc. 2nd Workshop on Economics of Peer-to-Peer Systems*, 2004.

[14] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer. KARMA: A Secure Economic Framework for P2P Resource Sharing. In *Proc. Workshop on Economics of Peer-to-Peer Systems (P2PECON)*, 2003.

[10] See CacheLogic Press Release http://www.cachelogic.com /home/pages/news/pr070806.php.

[11] See http://bittorrent.org/fast_extensions.html.

Capturing Complexity in Networked Systems Design: The Case for Improved Metrics

Sylvia Ratnasamy
Intel Research

ABSTRACT

The systems and networking community lays great store by "clean", "elegant" system designs. Yet, our notion of what these terms mean often relies more on intuition and qualitative discussion than rigorous quantitative metrics. This paper questions whether we can do better and takes a first stab at quantifying this notion of complexity with regard to the algorithmic component of a networked system design.

*While the success of our particular attempt is unclear, we believe identifying such metrics would be valuable not only in improving our own design and analysis methodologies but also to better articulate our design aesthetic to other communities that design for Internet contexts (*e.g., *algorithms, formal distributed systems, graph theory).*

1 INTRODUCTION

The design of a networked system frequently includes a strong algorithmic design component. For example, solutions to a variety of problems – routing, distributed storage, multicast, name resolution, data processing in sensor networks, resource discovery, overlays – all define distributed procedures by which a collection of nodes accomplish a network-wide task.

A much valued property in Internet systems such as the above is that of design simplicity. However, as the literature reveals, our rationalization about the simplicity (or lack thereof) of design options is often through qualitative discussion or, at best, proof-of-concept implementation. What rigorous metrics we do employ tend to be borrowed from the theory of algorithms. These metrics however were intended to capture the overhead or efficiency of an algorithm and are at times incongruent with our notion of what makes for simple systems. For example, two of the most common metrics used to calibrate system designs are the amount of state maintained at nodes and the number of messages exchanged across nodes. However, most of us would consider flooding a simple although inefficient solution. Similarly, a piece of state obtained as the result of complex consensus or leader election protocol feels intuitively more complex than state that holds the IP address of a neighboring node.

We conjecture that this mismatch in design aesthetic contributes to the frequent disconnect between the more theoretical and applied research on networking problems. A good example of this is the work on routing. Routing solutions with small forwarding tables are widely viewed as desirable and the search for improved algorithms has been explored in multiple communities; *e.g.*, a fair fraction of the proceedings at STOC, FOCS, PODC and SPAA are devoted to routing problems. The basic distance-vector and link-state protocols incur high routing state ($O(n)$ entries) but are simple and widely employed. By contrast, a rich body of theoretical work has led to a suite of *compact* routing algorithms (*e.g.*, [1–4]). These algorithms construct optimally small routing tables ($O(\sqrt{n})$ entries) but appear more complex and have seen little adoption. Such discrepancies are even more common in the context of sensor networks where the difficulties of the operational environment render simplicity that much more valuable.

Note that this is not to suggest existing performance metrics aren't relevant or useful. On the contrary, all else being equal, solutions with less state or traffic overhead, are strictly more desirable. The point – or rather conjecture – here is that design simplicity plays a role in selecting solutions for real-world systems but existing performance-focused metrics can be incongruent with our notion of what constitutes elegant system design.

This paper raises the question of whether we can identify metrics that more directly capture the intuition behind our judgment of system designs. Some might view system design and evaluation as inherently reliant on the design aesthetic and experience of system designers. The conjecture behind this paper is that maybe this need not be true – the system designs we work with are sufficiently deterministic that there ought to be no fundamental reason why our appreciation of a design cannot be based on quantifiable measures. Such metrics, if we can identify them, would not only allow us to more rigorously discriminate between design options but also to better align the design goals of the algorithms and systems communities.

This paper takes a first stab at identifying such metrics. Our results are preliminary, intended primarily to initiate discussion on the merits and nature of alternate metrics. Moreover, we stress that our metrics are intended to complement, and not replace, existing performance metrics. For example, in the case of a routing algorithm, our metrics might capture the complexity of route construction but reveal little about the quality of computed paths. Finally, while we focus on system design at the algorithmic or procedural level, there are many aspects to a software system

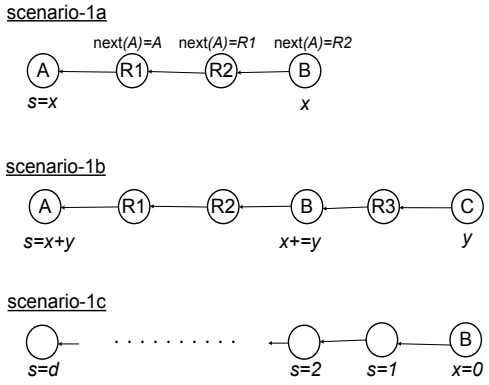


Figure 1: *Complexity in different scenarios.*

that contribute to its ultimate complexity. For example, as the CAP theorem [5] tells us, the very definition and prioritizing of a system's service model and guarantees, profoundly impacts complexity. The same is true for the sound design of its software implementation. Although at least as important as distributed complexity, these are not aspects we consider in this paper.

2 STRAWMAN

At a high level, one might view much of system design as centered around the issue of *state* – defining what state is required, how it is constructed and used by different operations, and so forth. For example, a routing solution defines the forwarding entries required, the process by which nodes discover these entries, and how a packet is delivered end-to-end using this forwarding state. In all this, the strain particular to wide-area systems arises when state is distributed and hence a given piece of state is dependent not only on the different nodes storing its input state but also the network and intermediate nodes needed to relay this input state to the node in question. In other words, for a given piece of state, not only are its dependencies distributed, there are also more of them. Moreover, relative to a centralized or cluster environment, these dependency-inducing elements (input/relay nodes, links, *etc.*) tend to be more independent in their failure or change models. While traditional metrics count the amount of state but otherwise mostly treat all state as equal, we postulate that a key ingredient to capturing the difficulties in a networked system is to measure the *ensemble of distributed dependencies* that must hold together for a given piece of state to be consistent with the inputs from which it is derived.

In what follows, we attempt to develop such a metric. Metrics are only as useful as they are usable and, it is worth noting that current popular metrics simply count the total state and messages. These are conceptually simple, and lend themselves to evaluation through simple examination, analysis, or even mechanistically in simulation. A key goal we set is to define metrics that are somewhat similarly accessible. Our strategy – at least in this first cut – is to limit ourselves to metrics that only involve *counting* the different dependencies and avoid incorporating intricate models of node or link failure, state machine descriptions and the like. We discuss some of the limitations of our counting-based approach later in the paper.

We use a series of incremental observations and toy scenarios to help develop our proposed metric. Our discussion considers only distributed dependencies in state and, where the context is clear, abuses notation to let state identify the node storing the state; *e.g.*, instead of saying delivered to node X that stores state s, we simply say delivered to s.

Value *vs.* transport dependencies: We start with the case where a piece of state, denoted s, is derived from a single input state, denoted x. For example, in scenario-1a in Figure 1, x denotes the current temperature reading at node B and state s at node A is assigned the value of x. The value of s is derived from x and hence any change in x must be communicated to node A. By contrast, s is dependent on the $next(A)$ state at B, $R1$ and $R2$ only for the delivery of x to s but a change in any of these does not require an update to s. We distinguish between these two forms of dependencies and say s is *value* dependent on x and *transport* dependent on $next(A)$, at B, R1 and R2.

Let v_s denote the number of pieces of state on which s is value dependent, and $t_{s \leftarrow x}$, the number of pieces of state relied on to transport x to s. Since we're only interested in distributed dependencies, we set $v_s = t_{s \leftarrow s} = 0$ if s was generated at the local node; thus, in Figure 1, $v_x = t_{x \leftarrow x} = 0$ and correspondingly, $v_s = 1$ and $t_{s \leftarrow x} = 3$. Note that a piece of state is not necessarily value dependent on its inputs. For example, say we defined s as the temperature at node B at a specific time T1 (as opposed to B's current temperature). In this case, once established, s is unaffected by changes at node B or the network between A and B. Thus, for s derived from x, we set $v_s = v_x + 1$ if s is value dependent on x and $v_s = max(v_x, \varepsilon)$ otherwise, where ε $(0 < \varepsilon \ll 1)$ is a minimal dependency value we introduce to ensure all non-local state has a non-zero dependency which also captures the one-time cost of state initialization. Similarly, to ensure than any inter-node communication incurs a minimal dependency cost, we set $t_{x \leftarrow y} = max(\varepsilon, t_{x \leftarrow y})$, for adjacent x and y.

Combining value and transport dependencies Consider the slightly more involved scenario-1b in which y records the current temperature at node C, x represents the sum of y and the current temperature at B, and s is once again set to x. Now, $v_x = 1$ (since x also depends on y) and hence $v_s = 2$. The transport dependency $t_{y \leftarrow y} = 0$, $t_{x \leftarrow y} = 2$, and $t_{s \leftarrow x} = 3$. We note that value dependencies accumulate in a fairly straightforward fashion but the extent or frequency with which transport dependencies $t_{s \leftarrow x}$ are incurred depends on the number of value dependencies downstream from x. For example, s depends on $t_{s \leftarrow x}$

to relay changes in the temperature at either B or C but changes in y are only relayed using $t_{x \leftarrow y}$.

Based on the above discussion, for state s derived from a single input x, we define c_s, the complexity of s as:

$$c_s = v_s \times t_{s \leftarrow x} + c_x$$

Thus, for s in scenario-1, $c_s = 3$ and $c_x = 0$ while in scenario-2, $c_y = 0$, $c_x = 2$ and $c_s = 2 \times 3 + 2 = 8$.

Note that c_s emphasizes the simultaneous importance of balancing both value and transport dependencies in achieving low complexity – a single dependency input x ($v_x = 1$) delivered to s via a convoluted network path is deemed complex as is an input that is one hop away ($t_{s \leftarrow x} = 1$) but x itself is derived from a long chain of previous inputs.

As a final example before proceeding, consider scenario-1c in Figure 1. Here $x = 0$ and each node computes s, its distance to x by incrementing its right-hand neighbor's value of s by 1. We abuse notation and let d denote a node with $s = d$; then we have $v_d = d$ and $t_{d \leftarrow (d-1)} = 1$ and $c_d = d \times 1 + c_{d-1}$. We have $c_x = 0$ and hence $c_d = O(d^2)$.

Multiple inputs So far, we considered s derived from a single input x. (Note that by input, we mean direct inputs; *e.g.*, in scenario-1b, we consider x as input to s but not y.) We now consider the case where s is derived from m inputs $x1$, $x2$, …, xm. We consider two basic variants that can be combined to yield more complex scenarios. In the first, *all* m inputs are required to compute s (*e.g.*, computing the min, max or average of m input readings); in the second s can be derived from *any* one of the m inputs (*e.g.*, recording liveness). For simplicity, we assume $t_{s \leftarrow x_i} = 1$ in both cases.

When all m inputs are required, we set:

$$v_s = \sum_{i=1}^{m} (v_{xi} + 1)$$

$$c_s = \sum_{i=1}^{m} ((v_{xi} + 1) \times t_{s \leftarrow xi} + c_{xi})$$

Thus if $v_{xi} = c_{xi} = 0$, we have $v_s = c_s = m$.

In the second scenario, s can get by with any one of m inputs coming through. Accordingly, we set the value dependency and complexity of s as follows:

$$v_s = \frac{1}{m} \times \sum_{i=1}^{m} (v_{xi} + 1/m)$$

$$c_s = \frac{1}{m} \times \sum_{i=1}^{m} ((v_{xi} + 1/m)\ t_{s \leftarrow xi} + c_{xi})$$

Note that the above reflect the observation that in the one-of-m variant, s is less dependent on each individual input and does not depend on the sum total of all inputs. Again, when $v_{xi} = c_{xi} = 0$, we have $v_s = c_s = 1/m$.

Case	Description	v_s	c_s
1	s=x; s,x are 1 hop apart	1	O(1)
2	s=x; s,x are k hops apart	1	O(k)
3	s=hops to x; s,x are k hops apart	k	O(k^2)
4	s=ALL(x1,…,xm); s, xi 1 hop apart	m	O(m)
5	s=ANY(x1,…,xm); s, xi 1 hop apart	$1/m$	O($1/m$)
6	s=x; m 1-hop paths from x to s	1	O($1/m$)

Table 1: *The value dependency and complexity for a single piece of state s for various base-case scenarios.*

Multiple paths There may exist multiple paths by which an input can be delivered to the required node. For example, consider the case where x is delivered to s along any one of m disjoint paths, and the transport dependency of each disjoint path is (say) d. We treat multiple paths akin to the corresponding multiple input scenarios and hence set $t_{s \leftarrow x} = d/m$ and hence the complexity $c_s = d/m$ which is lower than the single-input-single-path case by a factor m. [1]

Table 1 summarizes our complexity evaluation for state s in the various toy scenarios. We see that, as one might expect, the complexity of state derived from a single input (case#1) is less than that derived from m inputs (#4) but greater than for one-of-m-inputs (#5). Similarly complexity decreases as the network offers more delivery options between input and output (1 vs. 6). Note too that, our metric penalizes a value dependency of d that is accumulated in series or depth (#3) more than the same value dependency accumulated in breadth (# 4). This is reasonable as deeper dependencies incur more transport dependencies.

Operations on state So far we looked at computing the complexity of a given piece of state. A similar strategy can be used to compute the complexity of an operation – we treat the pieces of state the operation acts on as inputs and, based on how these inputs are combined, compute the operation's complexity from the individual state complexities. *E.g.*, a packet forwarding operation destined for D relies on the routing table entry for D at each of the series of nodes from the source to D. Specifically, recall the previous scenario-1c, where each node learns its distance and next hop to x (node B). We had computed the complexity of state d hops away as O(d^2). Forwarding a packet from A to B requires the state at each intermediate node for a complexity of O(d^3) (sum of squares). Or, consider a file download that takes one of m inputs where each in-

[1] Many reviewers remarked that the decision to treat one-of-m as having a factor m lower complexity than the case of a single input/path is somewhat debatable because, ultimately, one of the inputs/paths is made use of. While this is a valid point that merits further scrutiny, the rationale behind the current choice is that a piece of state is less dependent on a single input if alternate inputs are easily available although this reasoning might be conflating simplicity and robustness. Note too that, while the complexity of a single piece of state (in the one-of-m case) may be lower, the cost of creating more state for the purpose of redundancy will emerge in consider the net complexity of the complete system.

put is a pointer to a replica for the file. If each input has a complexity of (say) O(k) then, akin to the one-of-m inputs case, our download operation has complexity O(k/m).

In the following section we present some preliminary analysis of more complete networked designs. However, before doing so, we discuss some of the limitations of our proposed metrics and possible improvements.

2.1 Limitations, Future Directions

Correlated inputs: Our formulation treats inputs as independent and hence might be over-counting the dependencies. For example, the above forwarding operation sums the state complexities at each hop even though these are related. The extent of inaccuracy this introduces as well as compensating measures remains to be studied.

Discriminating between transport dependencies : Our formulation merely counts the number of transport dependencies however each transport dependency is itself state with its own value dependency and complexity and taking these into account might lead to more discriminating metrics. For example, we might instead sum the value dependencies of each transport state.

Capturing absent dependencies Our formulation measures the complexity involved in having state be consistent with the inputs from which it is derived. However, this does not necessarily capture *all* the dependencies that cause the state to take the value it does. For example, we measured the complexity of finding the distance k between two nodes s and x. However, this value of k depends as much on the *absence* of nodes between s and k that could lead to a different value of k as it does on the presence of the $k-1$ nodes between s and x.

Robustness vs. Simplicity Our formulation assigns lower complexity to state derived along alternate inputs/paths and hence reflects robustness to some extent. This link is however indirect and potentially limited; clearly relating complexity to robustness is an important future direction. Related is whether it might be useful to discriminate across inputs based on the degree to which the ouput (whether state or operation) depends on each input. For example, a DHT route critically depends on the successor entries but the absence of appropriate finger entries only leads to route degradation.

Scope Our metrics do little to validate the assumptions, correctness or quality of a solution. Capturing notions of consistency and convergence might require incorporating a notion of time or temporal dependency into our formulation and is another avenue for future exploration.

3 Initial Complexity Studies

This section presents preliminary analysis of a few common networked systems. We offer a high-level sketch of results with no detailed derivations; our intention is more to offer concrete examples of the type of analysis one might undertake in this context. We explore classical routing solutions in Section 3.1, and, in Section 3.2, look at resource location solutions in the context of P2P and sensor networks.

3.1 Network Routing

Our first study compares the complexity of distance-vector (DV) and link-state (LS) to the compact routing algorithm of Abraham *et al.* [4] (`AG+_compact`) which probably represents the state-of-the-art in compact routing. For simplicity, our analysis assumes a single shortest path to a destination.

In the case of DV, the routing entry (denoted s) for a destination d hops away is akin to case-3 in Table 1 and hence $v_s = O(d)$ and $c_s = O(d^2)$ and an end-to-end forwarding operation has complexity $O(d^3)$. In LS, a node propagates its link information to every other node and hence the entry e for a single edge has $v_e = O(1)$ and $c_e = O(d)$ (because the transport dependencies are O(d)). To compute the actual next-hop entry (denoted s) for a destination, LS requires the state for each of the d edges to the destination and hence, once again, $v_s = O(d)$, $c_s = O(d^2)$ and end-to-end forwarding has complexity $O(d^3)$ akin to DV.

`AG+_compact` guarantees routing table sizes with O($\sqrt{n}$) entries and worst-case stretch no more than 3.0. Moreover, the stretch for Internet-like topologies has been shown to be ≈1.0 for Internet topologies [6], raising the question of whether compact routing might be an attractive option for IP routing. Briefly, `AG+_compact` operates as follows: a node A's vicinity ball (denoted VB(A)) is defined as the k nodes closest to A. Node A maintains routing state for every node in its own vicinity ball as well as for every node B such that A ∈ VB(B). A distributed coloring scheme assigns every node one of c colors. Under a slight relaxation this can be done by simply hashing the node name to a color. One color, say red, serves as the global backbone and every node in the network maintains routing state for all red nodes. Finally, a node must know how to route to every other node of the same color as itself. For n nodes, vicinity balls of size $k = O(\sqrt{n}\log n)$ and $c = O(\sqrt{n})$ colors, one can show that a node's vicinity ball contains every color. With this construction, a node can always forward to a destination that is either in its own vicinity, is red, or is of the same color as the node itself. If none of these is true, the node forwards the packet to a node in its vicinity that is the same color as the destination. The challenge in `AG+_compact` lies in setting up routes between nodes of the same color without requiring state at intermediate nodes of a different color and yet maintaining bounded stretch for all paths. Loosely, `AG+_compact` achieves this as follows: say nodes A and D share the same color and A is looking to construct a routing entry to D. A explores every vicinity ball to which it belongs (VB(I), A ∈ VB(I)) and that touches or overlaps the vicinity ball of the destination D (*i.e.*, ∃ node X

$\in$ VB(I) with neighbor Y and Y $\in$ VB(D)). For such C, A could route to D via C, X and Y. `AG+_compact` considers possible paths for each neighboring vicinity balls VB(C) as well as the path through the red node closest to D and uses the shortest of these for its routing entry to D. Discovering A's membership in a node B's VB itself incurs significant dependencies – unlike DV/LS where a node maintains distance for any and every unique destination it hears about, here B maintains state for A iff A is one of the k nodes closest to B. In other words, whether B maintains state for A depends on the relative distance of *other* nodes to B which already induces a dependency of at least O($\sqrt{n}$). Moreover, the construction of intra-color routing entries requires that A explore all vicinity balls in which it is contained, and those of each of the $O(\sqrt{n})$ like-colored nodes which yields a total dependency of $O(n)$ – significantly higher than DV or LS!

Such analysis offers hints for alternate designs. For example, we conjecture that one might reduce the above dependencies by $\sqrt{n}$ if we defined nodes' vicinity balls not as an ordering of nodes but in terms of the distance around each node; with this change, A's membership in B's vicinity ball would depend only on A, B and the nodes between them. While such a change would likely weaken the bounds on the size of routing tables it could offer lower complexity.

3.2 Resource Discovery

P2P resource discovery Many P2P applications locate resources using either unstructured (Gnutella) or structured (DHT) overlays. For the former, each node connects to some number of other peers and each neighbor entry s thus has $v_s = 1$ and $c_s = 1$. (This assumes a transport dependency of 1 for overlay links.) By comparison, a DHT node might have $\log n$ neighbors, each with $v_s = 2$ and $c_s = 2\log n$ (due to a value dependency of one for a node's successor and hence two for a finger entry; the transport dependency is $\log n$ ignoring once again multiple paths). The corresponding complexity of end-to-end DHT routing is thus O($\log^2 n$).[2] This would seem to support deployment statistics and the common perception that unstructured solutions are simple, if inefficient. In absolute terms though, DHTs too exhibit low complexity which again would seem to concur with the general enthusiasm for DHTs in the systems and networking communities.

Resource discovery in sensor networks By far the most common approach to resource location in sensor nodes uses a flood-and-find approach where a sink floods the query over the entire network and relevant data is routed along the reverse path to the sink [7, 8]. The per-node network state here is merely the parent to the sink which, akin to the simple distance counting scenario in Section 2, has $v_s = k$ and $c_s = O(k^2)$ where k is the node's distance to the sink. While k and O(k^2) appears fairly low complexity, it is worth noting that in a sensor network, k can be O($\sqrt{n}$) leading to non-trivial complexity; we conjecture this may offer some insight on the engineering difficulties that have been reported for even simple tree construction [8, 9] and speculate that solutions based on gossip-style protocols [10] might be one approach to avoiding such scaling in dependencies.

To avoid the inefficiency of flooding, researchers have explored the use of in-network rendezvous mechanisms. One highly scalable proposal uses geographic addressing and routing [11–13]. In traditional geo routing, a node requires only the geo positions of its physical (*i.e.*, in radio range) neighbors. This incurs very low complexity – for each neighbor entry s, $v_s = 1$ and $c_s = 1 \times \varepsilon = \varepsilon$ (as discussed in Section 2, neighbor discovery effected through blind broadcasts might be viewed as incurring negligible transport dependency). Unfortunately, the adoption of geographic techniques has been hampered by both, concerns about the cost, power consumption and usability of GPS technology, and because empirical studies have repeatedly shown that wireless connectivity is not always congruent with geo proximity violating a core assumption of geo-routing. [3]

Two research directions address these concerns. Schemes such as CLDP [14] and GDSTR [15] continue to require GPS but propose novel route recovery algorithms that tolerate incongruities between physical distance and connectivity. The second approach eschews the use of geography altogether; schemes such as GEM [16] and NoGeo [17] instead construct virtual coordinate systems derived from only the measured connectivity between nodes. Like traditional geo routing algorithms, these new schemes are scalable in terms of the routing state but require additional mechanisms to either recover from route failures (CLDP, GDSTR) or to construct the virtual coordinate system (GEM, NoGeo). One might ask how much of the simplicity of traditional geo-routing is lost due to this? A quick analysis of the NoGeo protocol suggests that a routing entry s has $v_s = O(n)$ and $c_s = O(n^{3/2})$ (due mostly to a periodic initialization phase to position O($\sqrt{n}$) perimeter nodes). While the various schemes should be explored in greater depth, the above suggests a significant increase relative to both flood-and-find and the idealized promise of geo routing.

4 Discussion and Future Studies

Validating the goodness of a metric is, almost by definition, difficult and perhaps the best is to analyze a range of systems and examine the results. We close with a list of

[2] This DHT analysis may be overly generous as a node A's successor state is actually dependent not just on the identifier of A's successor but on the absence of any other node between A and its current successor; our metric does not currently capture such absent dependencies.

[3] Highlighting that complexity metrics do little to validate the assumptions behind a solution.

open questions and analyses that could help in this regard as well as offer insight on common design practices.

Centralizing network computations: simpler? Architectures that centralize the route computation have been proposed as a simpler alternative to today's distributed architecture [18] and it would be useful to undertake a formal analysis comparing the two. We conjecture the answer may depend on whether the value dependencies are "reset" at the centralized computation point. *I.e.*, on whether the final forwarding entries pushed to routers need to be consistent with the view of the world at the centralized route computation point or the true state of the world.

Designing for low dependency Section 2 presented a simple example where time-stamping temperature readings truncates the value dependency of the state being propagated. To some extent, soft-state protocols employ a somewhat similar strategy by bounding the lifetime of state and hence the length of dependencies it induces. Similarly, introducing redundancy in both inputs and transport dependencies lowers our measure of complexity. A useful exercise would be to quantify the complexity of systems that employ such techniques and verify whether their complexity matches our intuition.

Layered *vs.* **customized solutions** Some DHT applications [19–21] adhere to a standard DHT API and layer more complex functionality over this API while others [22, 23] choose to customize their DHTs to the task at hand. On the one hand, layering might lead to more needlessly inherited dependencies while the latter might introduce more mesh-like dependencies. In this context, one might for example compare the complexity of a CDN over a "sloppy DHT" interface [22] *vs.* the standard DHT interface or Mercury [23] that builds a customized solution to distributed range queries versus PHTs [21] that adopts a layered approach.

Network addressing and routing options A number of very different approaches to routing and addressing have been proposed in the context of both wireless and wired networks – gossip [10], synthetic coordinate systems [16, 17], clustering/dominating sets [24], tree-based [7,8], DHT-inspired [25] and so forth – that could be compared in terms of a more complexity-focused evaluation.

5 Acknowledgments

The author thanks Kevin Fall, Paul Francis and the anonymous reviewers for their valuable feedback.

References

[1] L. J. Cowen. Compact routing with minimum stretch. In *ACM SODA*, 1999.

[2] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg. Compact distributed data structures for adaptive routing. In *ACM STOC*, 1989.

[3] Mikkel Thorup and Uri Zwick. Compact routing schemes. In *ACM SPAA*, 2001.

[4] I. Abraham, C. Gavoille, D. Malkhi, N. Nisan, and M. Thorup. Compact name-independent routing with minimum stretch. In *16th ACM SPAA*, 2004.

[5] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent available partition-tolerant web services. In *SigACT News*, 2002.

[6] D. Krioukov, K. Fall, and X. Yang. Compact Routing on Internet-like Graphs. In *IEEE Infocom*, 2004.

[7] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: a scalable and robust communication paradigm for sensor networks. In *MOBICOM*, 2000.

[8] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad hoc sensor networks. In *OSDI*, 2002.

[9] Cheng Tien Ee, Sylvia Ratnasamy, and Scott Shenker. Practical data-centric storage. In *NSDI*, 2006.

[10] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickles: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *NSDI*, 2004.

[11] F. Kuhn, R. Wattenhofer, Y. Zhang, , and A. Zollinger. Geometric ad-hoc routing: Of theory and practice. In *22nd ACM PODC*, 2003.

[12] B. Karp and H. T. Kung. Greedy Perimeter Stateless Routing for wireless networks. In *MOBICOM*, 2000.

[13] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks.

[14] Y. J. Kim, R. Govindan, B. Karp, and S. Shenker. Geographic routing made practical. In *NSDI*, 2005.

[15] B. Leong, B. Liskov, and R. Morris. Geographic routing without planarization. In *NSDI*, 2006.

[16] J. Newsome and D. Song. GEM: Graph embedding for routing and data-centric storage in sensor networks without geographic information. In *Proceedings of SenSys*, 2003.

[17] Ananth Rao, Sylvia Ratnasamy, Christos Papadimitriou, Scott Shenker, and Ion Stoica. Geographic routing without location information. In *MOBICOM*, 2003.

[18] M. Caesar, D. Caldwell, N. Feamster, Jennifer Rexford, Aman Shikh, and J. Merwe. Design and Implementation of a Routing Control Platform. In *NSDI*, 2005.

[19] Matthew Harren, J. Hellerstein, Ryan Huebsch, B. Thau Loo, Scott Shenker, and Ion Stoica. Complex queries in DHT-based Peer-to-peer networks. In *IPTPS*, March 2002.

[20] F. Dabek et al. Wide-area cooperative storage with CFS. In *ACM SOSP*, October 2001.

[21] Yatin Chawathe et al. A case study in building layered DHT applications. In *Proceedings of SIGCOMM*, 2005.

[22] M. Freedman, E. Freudenthal, and D. Mazieres. Democratizing content publication with coral. In *NSDI*, 2004.

[23] A. Bharambe, M. Agrawal, and S. Seshan. Mercury: Support scalable multi-attribute range queries. In *SIGCOMM*, 2004.

[24] J. Gao, L.J.Guibas, J .Hershberger, L. Zhang, and An Zhu. Discrete mobile centers. In *Proceedings of the Symposium on Computational Geometry*, 2001.

[25] Matthew Caesar *et al.* Virtual Ring Routing: Network routing inspired by DHTs. In *SIGCOMM*, 2006.

Discovering Dependencies for Network Management

Paramvir Bahl, Paul Barham, Richard Black, Ranveer Chandra, Moises Goldszmidt,
Rebecca Isaacs, Srikanth Kandula[†], Lun Li[‡], John MacCormick, David A. Maltz, Richard Mortier,
Mike Wawrzoniak[¶], Ming Zhang.
Microsoft Research; also [‡] Caltech, [†] MIT and, [¶] Princeton.

This paper presents the Leslie Graph*, a simple yet powerful abstraction describing the complex dependencies between network, host and application components in modern networked systems. It discusses challenges in the discovery of Leslie Graphs, their uses, and describes two alternate approaches to their discovery, supported by some initial feasibility results.*

1 Introduction

It is lamentable that Leslie Lamport's famous quote [9] *"A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable"* describes a scenario familiar to almost every computer user. As IT systems are increasingly distributed, it is not only the clients and servers themselves that can render a computer useless for an afternoon, but any of the many routers, links and network services also involved.

In distributed systems, the underlying problem is the absence of tools to identify the components that "can render your own computer unusable": the implicit web of dependencies among these components exists only in the minds of the human experts running them. The complexity of these dependencies quickly adds up, requiring more help than traditional IT management software provides. Listing the contents of a single DFS [1] directory, for example, can involve a minimum of three hosts and eight network services (WINS, ICMP Echo, SMB, DFS, DNS, Kerberos, ISA key exchange, ARP). Existing management solutions focus on network elements, topology discovery, or particular services, but what is needed are tools to manage and improve the user's end-to-end experience of networked applications.

In deference to Lamport, this paper defines the *Leslie Graph* as the graph representing the dependencies between the system components, with subgraphs representing the dependencies pertaining to a particular application or activity. Nodes represent the computers, routers and services on which user activities rely, and directed edges capture their inter-dependencies. Different versions of a Leslie Graph can express different granularities of dependence for an activity — for some analyses, an Leslie Graph capturing inter-machine dependences at the granularity of IP addresses might be sufficient, while for others an Leslie Graph capturing inter-service dependencies at the granularity of software processes might be desirable.

This paper makes three contributions: (*i*) we define Leslie Graphs and discuss the challenges in finding them, (*ii*) we suggest important problems that Leslie Graphs could help solve, and (*iii*) we describe two ongoing projects that are exploring different approaches to automatically infer the Leslie Graphs.

[1] Windows Distributed File System

1.1 Existing Approaches

It might seem that the Leslie Graph for an application could easily be constructed if its designer generated rules that spell out the application's dependencies. Indeed, a number of commercial products such as MAM[2] and the DSI "System Definition Model"[3] do just this. However, this approach has several problems: the system could evolve faster than the rules; deployment of various forms of middlebox (e.g., firewalls, proxies) can change the application's dependencies without the rule writers even being aware; and rules are unavailable for legacy systems.

Similarly, analysis of configuration files to determine the Leslie Graph is insufficient as many dependencies among components are dynamically constructed. For example, web browsers on enterprise networks are often configured to communicate through a proxy, sometimes named in the browser preferences but frequently contacted through automatic proxy-discovery protocols that themselves rely on resolution of well-known names.

Systems have been proposed to expose dependencies by requiring all applications to run on a middleware platform instrumented to track dependencies at run-time [1, 4, 7]. However, heterogeneity defeats most such efforts in practice. Networks run a plethora of platforms, operating systems, and applications, often from a wide range of vendors. While a single vendor might instrument their software, it is unlikely that all vendors will do so in a common fashion; similarly, building all distributed applications over a single common middleware platform is infeasible. Furthermore, many underlying services on which others depend are legacy services and cannot easily be instrumented or ported to run over an instrumented layer.

1.2 Challenges Finding the Leslie Graph

In contrast to the above approaches, but also without explicitly defining some notion of a Leslie Graph, others have argued that a promising approach to inferring dependencies is to observe externally visible behavior of system components without parsing the contents of packets they send—the "black-box" approach [2, 13]. We follow this general approach, relying mainly on correlation of observed network

[2] http://www.mercury.com/us/products/business-availability-center/application-mapping/

[3] http://www.microsoft.com/windowsserversystem/dsi/sdm.mspx

traffic to infer system dependencies, and augmenting as required with other techniques such as active probing. However, there are several challenges with this approach.

False positives. The Leslie Graph is expressed in terms of dependency between components, which requires understanding their *causality*. However, using observed traffic results in measuring their *correlation*, which is not the same. For example, it is perfectly possible for unrelated conversations, such as periodic background maintenance traffic, to exhibit misleading timing correlations.

False negatives (caching). Statistical correlations require a substantial number of observations in the presence of noise (unrelated background traffic). However, much critical control plane and session setup behavior occurs relatively rarely. For example, it is difficult to determine that a web-browser's use of HTTP depends on both DNS and ARP through traffic observation alone, as the services are typically invoked once and the results cached.

Granularity. Many modern IT deployments use clusters of servers to implement load-balancing and resilience for critical tasks. Alternatively, in smaller systems multiple services will be hosted as separate processes on a single server. Thus, a vertex of the Leslie Graph might need to represent other than a single computer: at some times an entire cluster of computers will be appropriate, at others a single process on a single computer.

Complexity. Enterprise networks use a wide variety of applications [11], including complex services like authentication (Active Directory, IPSEC, Kerberos, RADIUS), remote file systems (AFS, DFS, NFS, SMB), web applications (Sharepoint, Wikis), communications (VoIP, IM, email) and utilities (printing, DHCP, ARP). The inter-dependencies between these are extensive and poorly specified.

Trust. To compute the Leslie Graph hosts must share information about their activities and are expected to do so truthfully. In an enterprise network this trust can be established and enforced by the company's network policy and administration procedures.

We restrict our subsequent discussion to discovery of Leslie Graphs in enterprise networks precisely because the latter two challenges, complexity and trust, make enterprise networks both a useful and feasible place to do so. We assume that we can place agents on a reasonable fraction of the computers on the network to monitor packets sent and received. These agents can also be used for tomography: taking measurements and enabling probing from many vantage points to discover network topology and resolve ambiguities in the Leslie Graph.

2 Leslie Graphs and Their Uses

Studies show that $\sim$ 70% of enterprise IT budgets are spent on maintenance.[4] The ability to create an enterprise's Leslie Graph could have a major financial impact by enabling the following techniques for management and troubleshooting.

Fault localization. A common source of frustration for users is when an application temporarily hangs for no readily apparent reason. The hardest part of resolving such problems is often locating the problem in the first place. Is it in an overloaded server? A policy configuration? A failed router or link? The Leslie Graph for an application not only summarizes the components that are involved, but also allows information from multiple clients to be combined to pinpoint faults through tomography.

Reconfiguration planning. A classic tale of unexpected consequences [10] involves an old machine configured to backup an SQL database. Since the dependency of the primary server on this old machine for backup service was not explicit, operators re-imaged and recycled the old machine. Unfortunately, the primary server failed around the same time, and the database was completely lost. Companies are continually adding, reorganizing, or consolidating services. Frequently, changes are disruptive to services beyond those directly involved due to unexpected and previously hidden interactions. Planning these changes and diagnosing the problems that inevitably result is expensive.

Leslie Graphs can be expected to help in two ways. First, by automatically detecting dependencies, unexpected consequences can be identified in advance and planned for. Second, Leslie Graphs allow IT departments to warn ahead of time the users who will be affected by changes.

Helpdesk optimization. The fact that many users are active at the same time means that failures are likely to result in many calls to the helpdesk — initiating a new diagnostic effort for each call would be wasteful. Knowing the dependencies among components means that new reports can be rapidly chained to the trouble ticket of a known issue: eliminating time spent investigating dependent issues. It also reduces the likelihood of inappropriate remediation such as unnecessarily rebooting the user's computer, and it helps to prioritize trouble tickets by the numbers of users affected.

Anomaly detection. If Leslie Graphs are automatically constructed based on the observed behavior of hosts, anomalies and changes in the graphs point to hosts that are worthy of more detailed human investigation. For instance, differences between clients can be used to find policy issues. If a set of clients cannot reach a server while everything is fine for another set of clients, our algorithms will localize the problem to the clients. The structure of the Leslie Graph can then help guide a human to determine if the cause is a middlebox/firewall common among the clients or a policy (e.g., IPSEC) on the clients themselves.

[4] Forrester Research, "Governing IT in the enterprise" (July 2004) and http://research.microsoft.com/events/snmsummit

3 Implementation Considerations

We are exploring two different approaches to approximating the Leslie Graph using low-level packet correlations. The Constellation system uses a distributed approach, reactively constructing the Leslie Graph of any node on-demand. In contrast, the AND system, which stands for Analysis of Network Dependencies, proactively maintains the approximate Leslie Graph at a centralized inference engine. The rest of this section describes these systems in more detail.

3.1 Constellation

In the Constellation system, local traffic correlations are inferred by passively monitoring packets and applying machine learning techniques. The basic premise is that a typical pattern of messages is associated with accomplishing a given task. Therefore, it is possible to approximate the Leslie Graph by taking the transitive closure of strongly correlated nodes, and furthermore we can detect or diagnose faults by observing the *absence* of expected messages.

In order to explore the class of machine learning approaches that are applicable, we have formalized the problem. Space precludes a complete presentation, but the following three concepts are critical:

Channel. A channel represents the entities between which messages flow and thus between which an edge exists in the Leslie Graph. For example, all packets sharing the same source and destination address might be designated as belonging to a single channel. Alternatively, at a finer granularity we might additionally use application protocol to identify a channel. Channels are described as *input* or *output* channels based on whether they represent messages received at or transmitted by a host.

Activity pattern. We assign a value of either *active* or *inactive* to each channel in the network over some fixed time window. A set of such assignments to channels at a node is an activity pattern for that node, indicating whether or not a packet was observed on each channel during the observation time window.

Activity model. The activity model for a node is a function mapping the activity pattern of the input channels to a vector of probabilities for each output channel being active.

The idea is that by repeatedly observing whether an output channel is active for a given input activity pattern, we can learn the activity model on a host. To do this we are investigating a number of alternative mechanisms including Naive Bayes Classifiers [8] and Noisy-OR models [12]. Our results so far show promise, but have also highlighted some of the inherent trade-offs for this approach. Since activity patterns discard all packet timings and counts within the observation window, picking a suitable duration for the window is critical. Over a very long time window we will learn that all channels are related, whereas selecting a window size that is too small will cause correlations to be missed. We are tackling this problem by building activity models simultaneously for a range of window size and working on good ways to combine the resulting models.

Constellation uses activity models on hosts to approximate Leslie Graphs in a completely distributed manner. The correlation coefficients in the activity model encode the confidence level for a dependency between two nodes. When a host wishes to learn its Leslie Graph for a particular service, it queries its relevant peers to find strong next-hop correlations in their activity models for when only the input channel on which the query was sent is active. This query is then forwarded to those peers who repeat the process, and the resulting transitive correlations combine to give a Leslie Graph from the point of view of the local host. When the Leslie Graph is large this has the advantage that we can order the search by "most likely" path. Leslie Graphs are generated *on-demand* and give a snapshot of recent history at each member host.

One of the challenges when combining local activity models to form a Leslie Graph is choosing an appropriate threshold for deciding that a correlation is strong enough to be part of the graph. At some correlation value for a given edge there is insufficient evidence to assume a causal relationship, and so the edge should be excluded. We are currently investigating this and several other issues, including statistical hypothesis tests for detecting both anomalous and normal changes to an activity model.

3.2 AND

The AND system consists of a centralized *inference engine* and a set of *agents*, one running on each desktop and server. Each agent performs temporal correlation of the packets sent and received by its host and makes summarized information available to the engine. The inference engine serves as an aggregation and coordination point: assembling the Leslie Graph for applications by combining information from the agents; ordering agents to conduct active probing as needed to flesh out the Leslie Graph or to localize faults; and interfacing with the human network managers.

Computing the Leslie Graph. Using the terminology of Section 3.1, we define a *channel* as a 3 tuple of [RemoteIP, RemotePort, Protocol]. Each agent then continuously updates a matrix of the frequency with which two channels are active within a 100 ms window. [5]

To construct the Leslie Graph, the inference engine polls the agents for their matrices. Figure 1 illustrates how aggregating matrices from multiple agents over a long period of time can find dependencies that might be obscured by caching, since even infrequent messages to a server become measurable when summed over many hosts. For example, many hosts will have a matrix similar to $H3$'s that shows

[5] We have found values from 100 ms to 1 s produce the same dependency graphs on clients, but servers that are heavily loaded may cause dependences to be spread over a larger time window.

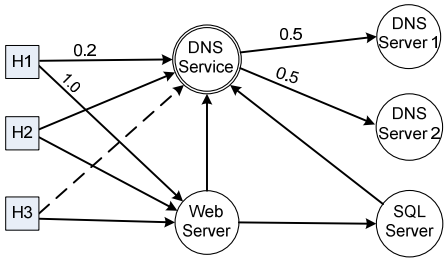


Figure 1: Part of a Leslie Graph discovered by AND when clients access some web server. The dashed line indicates a dependency found by aggregating information across hosts, $H1, H2, H3$.

a strong dependence on the web server, but no dependence on DNS as the web server's address has been cached. However, the matrices for $H1$ and $H2$ show that when these hosts communicated with the web server they also communicated with DNS in the same 100 ms window. If enough hosts that communicate on channel A (e.g., the web server) also communicate on channel B (e.g., DNS) within the same 100 ms, then the engine infers that any host depending on A most likely depends on B as well and will add to the Leslie Graph a dependency on B, as shown by the dashed line in the figure. Each edge in the Leslie Graph also has a weight, which is the probability with which it actually occurs in a transaction. In Figure 1, for example, $H1$ contacts the DNS server 20% of the time before it accesses the web server.

Networks that include either fail-over or load-balancing clusters of servers (e.g., primary/secondary DNS servers, web server clusters) are modeled by introducing a meta node into the Leslie Graph to represent each cluster, for example, the DNS Service node in Figure 1. Currently we use heuristics based on DNS names, port numbers, and stemming URLs to identify clusters and leave automatic detection of cluster configurations for future work.

In addition to user machines and application servers, AND extends the Leslie Graph by populating it with network elements, such as routers, switches and physical links. This broadens the applications of the Leslie Graph as, e.g., link congestion faults can now be localized. We can map the layer-2 topology by using the agents to send and listen for flooded MAC packets as in [5], and the layer-3 topology using traceroutes. Other techniques [6] could be used if SNMP data is available.

Using the Leslie Graph. Of the scenarios described in Section 2, our current focus is on efficient fault localization. Each agent observes the experiences of its own host (e.g., measuring the response time between requests and replies). When a user on the host flags the experience as bad, the agent sends a triggered experience report to the inference engine. For example, a negative experience report might be generated when a user restarts their browser or hits a button that means "I'm unhappy now", or when automated parsing identifies that something wrong has happened (e.g., too many "invalid page" HTTP return codes). A small number of randomly selected positive experiences (e.g., the time to load a web page when the user did not complain) are sent to the engine every 300 s.

The engine batches experience reports from multiple agents and applies Bayesian inference to find the most plausible explanation for the experience reports (i.e., the minimum set of faulty physical components that would afflict all the hosts, routers and links with poor performance while leaving unaffected the components experiencing acceptable performance). Space prevents a full description, but although Bayesian models typically require training, initial results (Section 4) show the structure of the Leslie Graph and the number of viewpoints provided by agents cause the results to have little sensitivity to the training process.

Scalability. The use of a centralized inference engine clearly makes it easier to aggregate information, but it raises scalability concerns about CPU and bandwidth limitations. On a single CPU, our system localizes faults on a Leslie Graph of 160 nodes (see Section 4 for details on the experiment setup) within 200 ms, with the time growing linearly in the number of nodes in the Leslie Graph.

Back-of-the-envelope calculations show the bandwidth requirements are feasible even for large enterprise networks. Experience reports are about 100 B and are sent to the inference engine every 300 s by each agent. The full co-occurrence matrix is polled from each agent every 3600 s. Most hosts in our network use fewer than 100 channels (i.e., use < 100 servers), so the matrix is less than 100x100 floats. Even for an extremely large enterprise network with O(100,000) computers and O(10,000) routers/switches, this results in an average bandwidth of only 10 Mbps. Busy servers have much more than 100 channels, but compression can be used if needed.

3.3 Discussion

As two separate projects, Constellation and AND are exploring two different points in the design space of approximating Leslie Graphs. While both approaches compute Leslie Graphs by aggregating the activities of multiple nodes, their differences highlight how the overall design space can be broken down into three axes, namely timing, structure, and granularity.

The first axis in the construction of a Leslie Graph is the time when it is constructed. Constellation constructs the Leslie Graph reactively, while AND proactively maintains it at the inference engine. If the Leslie Graph is constructed reactively, it imposes little overhead on hosts. However, since nodes log packets over a short period of time, a reactive scheme might miss out on dependencies affected by cached state. For example, in Figure 1, a reactive scheme might not determine the dependency between H3 and DNS. Furthermore, by proactively maintaining the Leslie Graph, the inference engine can respond to faults even before they are detected by all the users.

The second axis is the structure of the system, i.e. whether the Leslie Graph computation is centralized or distributed. Constellation uses a distributed approach to compute the Leslie Graph, while AND computes it at the centralized inference engine. A distributed (unstructured) approach is more robust to network and machine failures that might affect connectivity to the centralized server, while a centralized approach is simpler and easier to manage. We are improving the fault tolerance of AND by implementing the inference engine as a distributed cluster of machines.

The third axis is the granularity of the Leslie Graph, as discussed in Section 1.2. The nodes in the Leslie Graph could be a cluster of servers, a particular machine or a process on a machine. Similarly, for network elements, a node could be end-to-end connectivity between machines, or all the routers and switches in the path. The analysis on a more granular Leslie Graph will be more precise, although it might add complexity to the algorithm and unnecessary detail to the results. Constellation represents hosts and processes in the Leslie Graph, while AND also includes the routers and switches.

A notable trend is the increasing popularity of peer-to-peer applications. These are designed to achieve reliability by dynamically changing the set of servers with which a client communicates based on the content being exchanged or congestion levels in the system. As a result, the Leslie Graph is not stable across long time-periods. A system like Constellation, with its on-demand creation of the Leslie Graph using only recent observations, will report the set of peers currently in use. AND, which aggregates information across time, may show a dependency on servers no longer in use.

Our schemes do have limitations. We do not expect our techniques to find servers that return incorrect answers unless these errors lead to performance or fail-stop problems. For example, if a DNS server holds the wrong IP address for a name and only one client looks up the name, our approach will help only if the Leslie Graph changes as a result. Even if many clients lookup the wrong IP address and thus are unable to establish a connection, our fault localization algorithm will only point to the clients — although human examination of the Leslie Graph would reveal the affected clients share a DNS server. Here our tools and the structure of the Leslie Graph may help human investigators, even if they cannot automatically find the root cause.

4 Initial Results

Existence of correlations. The first step in validating our approach is determining if there is detectable correlation between input channels and output channels that are known to be related, and no correlation between unrelated channels. If this were not true, then black-box techniques would be infeasible. Figure 2 shows the results of plotting the time

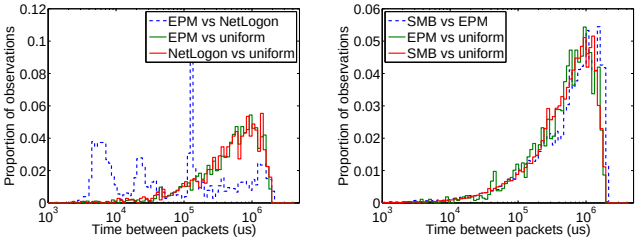


Figure 2: Evaluation of correlation between EPM packets & NetLogon packets (left) and EPM packets & SMB packets (right), using correlation with random noise as a control. EPM vs. NetLogon is significantly different from the control correctly validating their correlation while EPM vs. SMB is indistinguishable from the control.

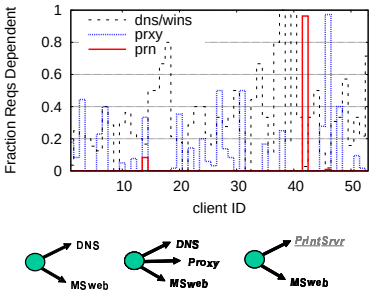


Figure 3: Example of finding dependencies of 53 hosts that contact an internal webserver. The figure on top shows the probabilistic dependencies discovered at each client. The figure below graphically represents the typical dependencies and also illustrates a false-dependency.

difference between receiving a packet of one protocol and sending a packet of the other protocol for three protocols: EPM (the RPC portmapper), NetLogon and SMB. Traces were collected for 1 hour from a busy server. The time differences are also computed against a packet stream whose timestamps are drawn from a uniform random distribution, representing background noise.

The right figure shows that SMB and EPM correlate as strongly with the random packet stream as they do with each other, implying they are not correlated. This is correct, as SMB and EPM are unrelated protocols. The left figure shows EPM and NetLogon have a very different distribution than the comparison with the random stream, implying EPM and NetLogon are closely related — in fact, NetLogon clients use EPM to locate the port to which they send their requests. We obtained similar validation for many other protocols, implying that packet-correlation techniques are feasible.

Finding dependencies. As a first test of AND's technique for finding dependencies in presence of caching, we ran the Leslie Graph generation algorithm against data from 53 hosts collected over one hour. Figure 3 shows the fraction of requests made by each client to the DNS, proxy, and PrintServer that co-occur with a request to a common webserver, called MSweb. As we can see, most clients invoke DNS when making web requests, although not 100% of the time due to caching. However, we still extract the

correct dependency. The data also show some clients are dependent on the proxy that is normally used for external access, even when accessing the internal web server. Investigation showed that these clients were misconfigured with an out-of-date list of internal names, indicating how our approach can be useful for detecting some classes of policy/configuration faults. Two misbehaving hosts made so many requests to the PrintServer that their dependencies for MSWeb became abnormal, showing that even false positives can yield valuable management information.

Usefulness of the Leslie Graph. To evaluate the ability of AND to find and use the Leslie Graph for fault localization, we have created a testbed with 23 clients that are evenly divided between two subnets connected by a router. Each subnet has a web server running Sharepoint (a wiki-like application), with data for the web sites stored on a single SQL database server on one of the subnets. Network services (DHCP, authentication servers, DNS) are connected to the subnet without the SQL server. Using packet-droppers, rate-shapers, and load generators we can deterministically create scenarios where any desired subset of the clients, servers, router, and links appears as failed or overloaded.

We evaluated five scenarios where combinations of one or more web servers, SQL server, routers, and links were set to an overloaded or failed state while robots on the clients made accesses to the web servers and each agent observed the response times seen by its client. These scenarios have Leslie Graphs with about 160 nodes, each of which is a component that could potentially fail. A small portion of the Leslie Graph for the testbed is shown in Figure 1. In all five scenarios, our fault localization algorithms run over the Leslie Graph correctly determined the problematic component. In three scenarios, the algorithm reported one more potentially problematic candidate than the number actually afflicted, but the algorithm also proposed the correct set of active probing tests to resolve this ambiguity.

5 Related Work

Project5 [2] proposes finding performance bottlenecks in a distributed system by using black-box approaches to track requests as they move between servers in the system. WAP5 [13] extends Project5 by developing a new message correlation algorithm for determining which arriving packets trigger which outgoing packets on a host. In contrast, this paper identifies the importance and challenges of discovering the Leslie Graph to support a broad range of management functions. By computing Leslie Graphs at different granularities, our techniques can uncover dependencies which might be overlooked by WAP5 and Project5, such as those masked by caching. We also present several new scenarios where the Leslie Graph can be applied. The notion of "Communities of Interest" (COIs) in enterprise networks is studied by Aiello *et al* [3]. A COI is defined more narrowly than the Leslie Graph, as a collection of interacting hosts, and the authors do not explicitly consider the problem of finding network dependencies.

6 Conclusion

As the web of dependencies between hosts, applications and network elements increases in size and complexity, building tools to automatically discover and reason about these dependencies will be invaluable for network operators and normal users. In this paper, we introduce the Leslie Graph as a generic representation of this web of dependencies. We present two complementary approaches to computing Leslie Graphs, and highlight the differences between them in three dimensions in the design space. We also present several applications of Leslie Graph and the challenges in discovering its approximation that have not been addressed in prior work. We are now gaining experience using the Leslie Graph, and so far have had success finding anomalous configurations and localizing performance faults, which tend to be transient, hard to debug, and annoying to users. We believe that the general problem of discovering and using Leslie Graphs presents a rich field of future research.

References

[1] A.Brown, G.Kar, and A.Keller. An active approach to characterizing dynamic dependencies for problem determination in a distributed environment. In *IFIP/IEEE IM*, May 2001.

[2] M. K. Aguilera, J. C. Mogul, J. L. Wiener, P. Reynolds, and A. Muthitacharoen. Performance debugging for distributed systems of black boxes. In *SOSP'03*, pages 74–89, Oct. 2003.

[3] W. Aiello, C. Kalmanek, P. McDaniel, S. Sen, O. Spatscheck, and J. V. der Merwe. Analysis of communities of interest in data networks. In *PAM'05*, Mar. 2005.

[4] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier. Using Magpie for request extraction and workload modelling. In *OSDI'04*, Dec. 2004.

[5] R. Black, A. Donnelly, and C. Fournet. Ethernet topology discovery without network assistance. In *ICNP'04*, Oct. 2004.

[6] Y. Breitbart, M. Garofalakis, B. Jai, C. Martin, R. Rastogi, and A. Silberschatz. Topology discovery in heterogeneous IP networks: The NetInventory system. *IEEE/ACM ToN*, 12(3), June 2004.

[7] M. Y. Chen, A. Accardi, E. Kıcıman, J. Lloyd, D. Patterson, A. Fox, and E. Brewer. Path-based failure and evolution management. In *NSDI'04*, pages 309–322, Mar. 2004.

[8] R. Duda, P. Hart, and D.G.Stork. *Pattern Classification*. Wiley-Interscience, 2nd edition, Oct. 2000.

[9] L. Lamport. Quarterly quote. *ACM SIGACT News*, 34, Mar. 2003.

[10] D. Oppenheimer, A. Ganapathi, and D. A. Patterson. Why do Internet services fail, and what can be done about it? In *USENIX SITS*, 2003.

[11] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney. A first look at modern enterprise traffic. In *IMC'05*, pages 15–28, Oct. 2005.

[12] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, Sept. 1988.

[13] P. Reynolds, J. L. Wiener, J. C. Mogul, M. K. Aguilera, and A. Vahdat. WAP5: Black-box performance debugging for wide-area systems. In *WWW'06*, May 2006.

Flexlab: A Realistic, Controlled, and Friendly Environment for Evaluating Networked Systems

Jonathon Duerig Robert Ricci Junxing Zhang Daniel Gebhardt Sneha Kasera Jay Lepreau
University of Utah, School of Computing

Abstract

Research prototypes of networked systems are often evaluated on overlay testbeds and emulation testbeds. Most of the strengths and weaknesses of these two types of testbeds are complementary. We outline the motivation, design, implementation, and sample results of an environment that seeks to provides the best of each type. Flexlab couples an emulation testbed with arbitrary network models. We also present a novel modeling technique tuned for this environment, *application-centric Internet modeling.* The key idea is to monitor the application's offered network load within the emulation testbed, replicate that load on the overlay testbed, measure the path's characteristics through analysis of the traffic, and use those to shape the emulated network.

1 Introduction

Emulation testbeds such as Emulab [25] and ModelNet [22] are valuable tools for understanding, testing and evaluating research prototypes of networked systems. They give users great control over host and network environments and offer easy reproducibility. However, emulation testbeds have a serious shortcoming: their network conditions are artificial and thus do not exhibit some aspects of real networks. Perhaps worse, researchers are *not sure* of two things: which network aspects are poorly modeled, and which matter to their application. We believe these are two of the reasons researchers underuse emulation environments; that emulators are underused has also been observed by others [23].

In this paper, we address this shortcoming by presenting Flexlab, a new testbed environment that enables a new generation of network models for emulation. We present three network emulation models built with Flexlab; they gather Internet measurements using PlanetLab. The first two use traditional measurement strategies, while application-centric Internet modeling is a novel technique for high-fidelity emulation. We rely on Emulab facilities to provide a friendly and controllable environment, but our techniques generalize to any emulator.

Currently, to get network conditions more realistic than those in an emulator, researchers use overlay testbeds such as PlanetLab [15] and RON [2], which provide a set of vantage points into the Internet. These testbeds also provide other orthogonal advantages: true geographic distribution, a service platform, and potential for deployment and real end-users. In this paper, however, we concentrate on their use as sources of realistic end-to-end network paths.

These live-network testbeds have some drawbacks that are not present in emulation testbeds. First, because of the popularity of overlay testbeds and the limited resources they possess, host resources such as CPU, memory, and I/O bandwidth are usually shared among many users and are frequently grossly overloaded, unrepresentative of typical deployment scenarios. Second, in today's overlay testbeds, users cannot perform many privileged operations, including choosing the OS, controlling network stack parameters, or modifying the kernel. Finally, overlay testbeds present a challenging environment for development, debugging, and evaluation [1, 19], three activities which represent a large portion of the work in networked systems research.

To combine the strengths of emulation testbeds (a rich, friendly, controllable environment) with the real network characteristics seen by live-network testbeds, Flexlab replicates these network characteristics inside of an emulator. Ideally, we would create a good *general-purpose* model of the Internet, and use that to drive the emulation, but this is an approach fraught with difficulties. A key obstacle is that a general-purpose emulator, in theory, has a stringent criterion for modeling accuracy: it must yield accurate results for *any* measurement of *any* workload. While much progress has been made on measuring and modeling aspects of the Internet for certain uses, such as improving overlay routing or particular applications [11, 12], creating good general-purpose models of the Internet is still an open problem [6, 7, 10]. Spring *et al.* [20] have made the argument that "reverse-engineering" the entire Internet over a 24-hour period is feasible. The limitation is that it would require an enormous community effort.

Given the difficulty of general-purpose modeling, we focus instead on the simpler problem of modeling the Internet as seen through the lens of an application. Flexlab does this by modeling end-to-end characteristics of Internet paths between pairs of overlay nodes. This reduces both the scale (hundreds of overlay sites vs. millions of Internet nodes) and the complexity (end-to-end measurement vs. detailed routing and queuing models) of the problem domain, making it more tractable.

The two simple models we present use measurements taken by general purpose measurement tools; the first sets static network parameters, and the second updates them

dynamically. Our third model, application-centric Internet modeling, takes a much higher-fidelity approach to network conditions: it measures the Internet using traffic similar to that generated by the application under test. This has the advantage that other traffic on the live network reacts similarly as it would to the application itself. It also removes any artifacts that might be introduced by special measurement traffic, and rare or transient network effects are immediately visible to the application. Finally, it is our belief that experimenters will be substantially more trusting of this concrete approach to modeling than to more abstract models.

These models are by no means an exhaustive set, but represent interesting points in the space of Internet models. Many other models are possible. For example, recent work [8, 11, 13] provides novel ways to trade off accuracy for decreased measurement traffic. In addition, models can be made *replayable*, using the same network parameters for multiple runs of an experiment. This enables repeatable experimentation, a feature not possible on the Internet.

Related Work. Network measurement to understand and model network behavior is a popular research area. There has been an enormous amount of work on measuring and modeling Internet characteristics including bottleneck-link capacity, available bandwidth, packet delay and loss, topology, and network anomalies; we cite only a few examples [5, 18, 17, 26]. In addition to use in evaluating protocols and applications, network measurements and models have been used for maintaining overlays [2].

The past few years have also seen growth of experimental network testbeds. Emulab and PlanetLab are the most prominent. The wide-area scope and realism of PlanetLab has attracted several measurement studies that are specific to it [19, 9, 27, 14]. Our work differs from these in its novel bridging of live-network experimentation and emulation.

2 PlanetLab Network & Host Conditions

In this section, we motivate Flexlab in two ways.

Scheduling Accuracy. Today's largest and most public overlay testbed, PlanetLab, is heavily overloaded. It does not represent a realistic deployment environment for many applications, so can cause significant accuracy problems. For example, our measurement agent detailed in Section 5 ran fine on unloaded nodes, but required significant optimization work to produce accurate results on PlanetLab. One of the ways this overload manifests itself is in scheduling jitter. To confirm this, we implemented a test program which schedules a sleep event and measured the actual sleep time using the system clock; the difference between the target and observed wakeup times indicates CPU starvation.

Our scheduling experiment ran on three PlanetLab nodes with differing load averages (roughly 6, 15, and 27) and an unloaded Emulab node running the same OS and kernel; the kernel schedulers run at 1000 Hz. 250,000 sleep events were continuously performed on each node with a target latency of 8 ms, for a total of about 40 minutes. Figure 1 shows the resulting CDF of the additional delay for the sleep events, up to the 90^{th} percentile.

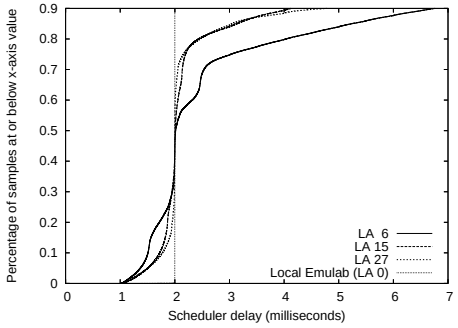


Figure 1: 90^{th} percentile scheduling time difference CDF

On the unloaded Emulab node, the CDF is reduced to a vertical line, due to equal sleep times. On the PlanetLab nodes, 90% of the sleep times are within 2–5 scheduler quanta (milliseconds) of the target, but there is a tail extending to several hundred milliseconds (99.99% of operations return in less than 150 ms). This tail is significant, and poses fidelity problems for programs that are time-sensitive. Many programs will still be able to obtain accurate results, but it is difficult to determine in advance which applications are sensitive to scheduling latency. The effect of scheduling jitter can also be reduced if, as Spring *et al.* [19] point out, tools can be designed so that untimely measurements are discarded. However, this is a difficult proposition for many programs.

Network Conditions and Stationarity. We find that the network conditions seen by PlanetLab can change frequently at small time scales. Importantly, we find the most variability on commodity Internet links, which, while being a minority in PlanetLab, comprise a majority of links in the Internet. To quantify this variability, we ran an experiment that collected high-frequency data on network latency. We do not claim that our experiment captures all interesting variation on these paths; the lesson to be learned from this experiment is that coarse measurement is not sufficient to capture all of the interesting detail of an Internet path.

Our experiment sent pings between pairs of PlanetLab nodes every 2 seconds for a 30 minute period, and analyzed the latency distribution to find "change points" [21]. Change points are points in time when the magnitude of the samples significantly changes; this statistical technique was used by a classic paper on Internet stationarity [28]. We use a method similar to their "CP/Bootstrap" test.

Table 1 shows some of the results. We used representative nodes in Asia, Europe, and North America. One set of North American nodes are on the commercial Internet, and the other are on Internet2. The third column shows the number of change points observed using all gathered data. The fourth column gives the magnitude of the median change for the path as a percentage of the median latency. The second

Path	20 sec. Period Count	2 sec. Period Count	2 sec. Period Size %
Asia to Asia	1	2	0.1
Asia to Commercial	0	2	2.9
Asia to Europe	0	4	0.5
Asia to I2	0	6	0.6
Commercial to Commercial	2	20	39.0
Commercial to Europe	0	4	3.4
Commercial to I2	1	13	15.0
I2 to I2	0	4	0.02
I2 to Europe	0	0	—
Europe to Europe	1	9	12.0

Table 1: Change point analysis for latency.

column is derived from the same data, but downsampled to 20-second intervals, simulating lower-frequency measurement. These results show that there are a number of paths that show significant variation at small time scales, and that low-frequency measurement misses nearly all of the change points in such data. The apparent discrepancy between our results and earlier studies, which found much less variation, is explained by our much higher measurement frequency.

3 Overall System Architecture

Flexlab's architecture is outlined in Figure 2. Because we concentrate on emulating end-to-end path characteristics rather than the full Internet topology, Emulab nodes are connected in a full mesh, abstracting the Internet as a set of pairwise network characteristics. An experiment can contain a mix of links modeled by Flexlab and traditional emulated links; each node to participate in a Flexlab link is associated with a PlanetLab node from which it will get its network characteristics. The experimenter can select specific PlanetLab nodes or allow Emulab to select node for them, based on the hosting site or Internet connectivity class (such as commodity Internet, Internet2, or non-North American).

The application under test runs on Emulab hosts and its network operations, such as `connect()` and `send()`, are recorded by the application monitor. The network model, which is easily "pluggable," feeds network parameters into the path emulator, a version of Dummynet [16] that we have enhanced with support for new path characteristics. The network model can also optionally use data from the measurement repository, which currently contains over three million low-frequency path measurements collected from PlanetLab over a period of several months. The model sends network parameters using Emulab's event system, which is a publish/subscribe system. Any node inside or outside of the experiment can publish new characteristics for paths; this makes it easy to implement either centralized or distributed model computation.

Most parts of this infrastructure are user-replaceable, allowing for a wide variety of models. We present three such models in the remainder of this paper; they are intended as beginning points in the exploration of pluggable network models rather than destinations. Our framework enables

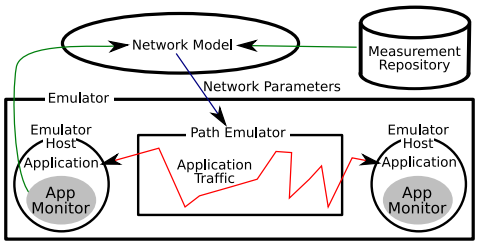


Figure 2: Flexlab Architecture.

transparent switching between different models, and even between Flexlab experiments and running live on PlanetLab. This eases the task of figuring out which network model is appropriate for an application, and enables development/debugging under simple, predictable models with evaluation done under more complex, realistic ones. This system is operational, and we have run a number of experiments using real applications on it.

We base our work on the Emulab network testbed management software, which provides important functionality. Emulab experiments may be interactive or completely scripted, and Emulab provides a distributed event system through which both the testbed software and users can control and monitor experiments. Emulab also provides efficient mechanisms for distributing experimenters' software to nodes, automatic packet trace collection, and gathering of logfiles and other results. Its "PlanetLab portal" [24] extends all of these benefits to PlanetLab nodes, allowing experimenters to easily move back and forth between emulation and live experimentation.

4 Simple Static & Dynamic Network Models

Our first two network models use data collected by our background monitor, which runs constantly on PlanetLab, taking measurements between all site pairs. Because there are a large number of pairs, this background monitoring can only be done at low frequency. We measure latency with `fping` every few tens of minutes, and assess bandwidth using `iperf` every few hours. (We found that less intrusive techniques such as packet-pair and packet-train were too imprecise and inaccurate on PlanetLab.) In the future we will reduce our need for active measurements by doing opportunistic passive measurements of file transfers by CoDeen, a popular CDN deployed on PlanetLab.

This low-frequency data is suitable for determining simple, static network parameters such as average latency and bandwidth. It is archived in the public Datapository [3] with which we federate, where it is available to any researcher. It can also be used by experimenters to choose paths that exhibit some desired characteristic, such as high variability or predictable diurnal variation. Our first network model, the "Simple-Static" model, uses this background information to set network parameters at the beginning of the experiment from historical data, and does not change them thereafter.

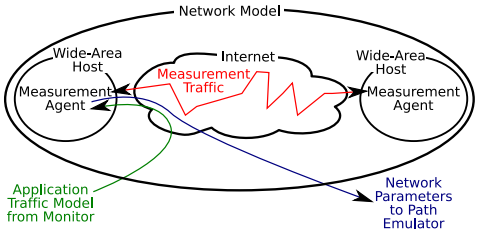


Figure 3: The data flow inside the application-centric Internet model.

As we saw in Section 2, real network conditions are dynamic, so this low-frequency data will not provide sufficient fidelity for many applications. For our next model, "Simple-Dynamic," Flexlab allows the experimenter or application to control measurement frequency, so that paths of interest can be monitored at higher frequency. As those new measurements are obtained, Flexlab continuously adjusts the path emulator parameters. This model uses information from the application monitor, which monitors which nodes actually connect to each other, so we can limit high-frequency measurements to these pairs.

In the future, an important use of the measurements archived in the Datapository will be replay: running emulations with network characteristics recorded in a previous run. With replay, researchers could repeat experiments with some variation of their application or its inputs. However, in such a replay, some applications may select different paths than they did during the original run, meaning that we may have only low-frequency data for those new paths. We can detect this situation, and give the experimenter feedback about the replay's fidelity.

5 Application-Centric Internet Modeling

Our desire to reproduce, with high fidelity, the network conditions that would be seen by an application run on PlanetLab, leads us to a technique we call application-centric Internet modeling. As discussed in Section 1, we do not attempt to trace, reverse-engineer, or model the full Internet. The key insight is that to produce a faithful model, we *need only model the Internet as perceived by the application*—as viewed through its limited lens.

The design of the application-centric Internet model is shown in Figure 3. Referring back to the Flexlab architecture in Figure 2, this is an instance of the "network model" component. The model receives characteristics of the application's offered load from the application monitor, replicates that load on PlanetLab with the measurement agent, determines path characteristics through analysis of the TCP stream, and sends the results back into the path emulator as traffic shaping parameters.

This design has several strong points. First, it creates a feedback loop in which we are constantly adjusting offered loads and emulator settings in near real-time. There is control latency in the communication between the emulator hosts and the wide-area nodes, but this merely time-shifts changes in offered load or network parameters by, typically, tens of milliseconds. Second, the design automatically obtains accurate information on how the network reacts to the offered load. Third, it lets us obtain fine-grained measurements of the traffic we send on PlanetLab, which allows us to track high-frequency network changes, such as we found in Section 2. Finally, it automatically and quickly detects the end-to-end effects of rare events such as outages and route flapping, which can be especially difficult to model.

We make some commonly-made assumptions about the Internet. We assume that most paths have a single bottleneck link, and that the location of that link does not change rapidly (though its characteristics may). We assume that ACK packets are not commonly dropped; missing ACKs are more likely due to forward path congestion. Finally, our work so far focuses only on TCP flows; we plan to extend it to UDP in the future.

5.1 Application Monitor and Measurement Agent

We pair each node in the emulated network with a peer in the live network. The application monitor, introduced in Section 3, runs on each Emulab node and connects to the measurement agent on the corresponding PlanetLab node. In turn, the agent sends updated network parameters to the appropriate path emulator in Emulab.

Application Monitor on Emulab. The applications under test are run with an `LD_PRELOAD` library which informs the monitor process of the application's network calls. Thus, any dynamically-linked executable can be instrumented without modification. The application monitor derives a model of the application's offered network load and sends this model to the measurement agent on the corresponding PlanetLab node. This model is simple and lightweight, consisting of the times and sizes of all `send()`s done by the application. The measurement agent can replicate the application's offered load by performing similarly sized and spaced `send()`s, albeit with different packet contents. By monitoring the application's offered load, rather than the packets it successfully sends on the wire, we see the data rate the application is *attempting* to achieve, rather than what it has been limited to by present network conditions. In some cases, this rate is artificially limited when the socket buffer is full. However, we still capture the true rate while it fills. The monitor also reports on important TCP settings, such as socket buffer sizes.

Measurement Agent on PlanetLab. Whenever the application running on an Emulab node connects to another node inside Emulab, the corresponding measurement agent on PlanetLab likewise connects to the agent on PlanetLab that represents the peer. The agent uses the load model sent by the monitor to generate similar network load, using `send()` calls, while also inspecting the resulting packet stream with `libpcap`. It collects fine-grained information on the TCP connection: for every ACK it receives from the remote agent, it calculates instantaneous throughput and

RTT. From these values it periodically generates and sends parameters to the path emulator in Emulab. To minimize PlanetLab host artifacts, the measurement agent requires little CPU time, and can distinguish between throughput changes due to available bandwidth and those caused by other effects, such as scheduling jitter.

5.2 Path Emulation

We emulate the behavior of the bottleneck router's queue within our path emulator, an enhanced version of the popular Dummynet [16] traffic shaper. The emulation uses two queues: a bandwidth queue, which emulates queuing delay, and a delay queue, which models all other sources of delay: propagation, processing, and transmission. Thus, there are three important parameters: the size of the bandwidth queue, the rate at which it drains, and the time spent in the delay queue. We assume that most packet loss in the wired Internet is caused by congestion, and thus induce loss only by limiting the size of the bandwidth queue.

Since the techniques in this section require that there be application traffic to measure, we bootstrap the model using historical data as in the Simple-Static model. These initial conditions will only be seen by the first few packets; after that, we have higher-quality measurements.

Bandwidth Queue Size and Packet Loss. When the bandwidth queue is full, arriving packets are dropped. The actual bottleneck router in the Internet has a queue whose maximum capacity is measured in terms of bytes and/or packets, but it is difficult to directly measure either of these capacities. Instead, we use a simpler approach: we approximate the size of the queue in terms of time. Sommers *et al.* [17] have proposed using the maximum one way delay to approximate the size of the bottleneck queue. This approach is problematic on PlanetLab because of the difficulty of synchronizing clocks, required to calculate one way delay. However, if we make the assumption that queuing delay along the reverse path does not fluctuate quickly, we can approximate the maximum queuing delay by subtracting the minimum RTT from the maximum RTT. We refine this number by finding the maximum queuing delay just before a loss event, yielding loss episodes consisting of both packets with high RTT and those that have been dropped.

Available Bandwidth. Measuring available bandwidth, the rate at which a flow's packets are drained from the bottleneck queue, has practical subtleties. Some measurement techniques do not take into account the reactivity of other flows in the network. For example, TCP's fairness (the fraction of the capacity each flow receives) is affected by differences in the RTTs of flows sharing the link, but measuring the RTTs of flows we cannot directly observe is difficult or impossible. We avoid these problems by directly measuring the bandwidth available to a specific connection, by sending that flow out into the network and measuring the resulting goodput, averaging it over the last half second to smooth outliers.

Deciding when to change the available bandwidth parameter in the path emulator has subtleties as well. If the application's offered load is not high enough to fully utilize the available bandwidth, we should not cap the bandwidth on the path emulator to this artificially low rate. Thus, we only *lower* the bandwidth available on the emulated path if we detect that we are fully loading the PlanetLab path. If we see a goodput that is higher than the goodput when we last fully utilized the path, then the available bandwidth must have increased, and we *raise* the emulator bandwidth.

Queuing theory shows that when a buffered link is overutilized, the time each packet spends in the queue, and thus the observed RTT, increases for each packet. Alternatively, we note that `send()` calls for a stream tend to block when the application is sending at a rate sufficient to saturate the bottleneck link. In practice, since both of these signals are noisy, we use a combination of them to determine when the bottleneck link is saturated. To determine whether RTT is increasing or decreasing, we find the slope of RTT vs. sample number using least squares linear regression.

Other Delay. The measurement agent takes fine-grained latency measurements. It records the time each packet is sent, and when it receives an ACK for that packet, calculates the RTT seen by the most recently acknowledged packet. We calculate the "Base RTT" the same way as TCP Vegas [4]; that is, the minimum RTT seen recently. This minimum delay accounts for the propagation, processing, and transmission delays along the path, with minimum influence from queuing delay. We set the delay for the delay queue to the Base RTT to avoid double-counting queuing latency, which is modeled in the bandwidth queue.

Outages and Rare Events. There are many sources of outages and other anomalies in network characteristics. These include routing anomalies, link failures, and router failures. Work such as PlanetSeer [27] and numerous BGP studies seeks to explain the causes of these anomalies. Our application-centric model has an easier task: to faithfully reproduce these rare events, rather than find the underlying cause. It automatically observes and mimics features of these rare events that are *relevant* to the application. Outages can affect Flexlab's control plane, however, by cutting off Emulab from one or more PlanetLab nodes. We plan to mitigate that by using an overlay network such as RON.

5.3 Sample Results

Figure 4 shows the throughput of a two minute run of `iperf`, which sends data as fast as possible over a TCP connection. The top graph shows throughput achieved by the measurement agent, which replicated `iperf`'s offered load on the Internet between AT&T and the University of Texas at Arlington. The bottom graph shows the throughput of `iperf` itself, running on an emulated path inside Emulab.

To induce a change in available bandwidth, we sent cross-traffic, in the form of 10 `iperf` steams, on the Internet path between time 35 and time 95. As we can see,

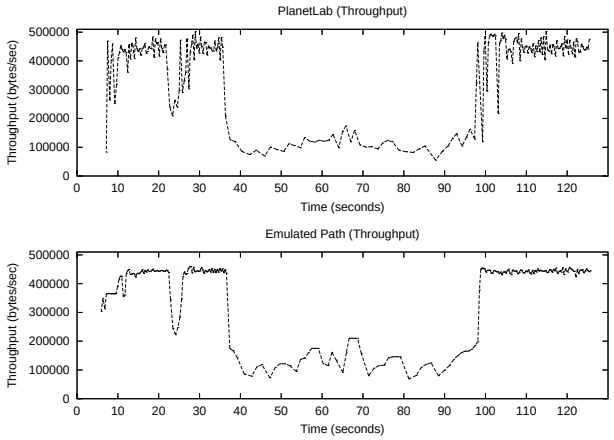


Figure 4: Application-centric Internet modeling, comparing throughput on PlanetLab (top) with the throughput of the application running in Emulab and interacting with the model (bottom).

Flexlab reacts quickly to the change, bringing the throughput of the path emulator down to the new level of available bandwidth. We next point out two other phenomena in this experiment. First, throughput drops in both streams around time 20; that change was presumably caused by cross-traffic from some external source. Second, brief but large drops in throughput occasionally occur in the Internet graph, such as those around time 100. These are due to the measurement agent not getting scheduled for extended periods and thus failing to saturate the link, demonstrating the artifacts due to scheduling jitter discussed in Section 2. The measurement agent correctly determines that these reductions in throughput are not due to available bandwidth changes, and deliberately avoids mirroring these PlanetLab host artifacts on the emulated path.

Acknowledgements

We are grateful to our co-workers for great help with implementation, evaluation, operations, design, and discussion: Kevin Atkinson, Russ Fish, Sachin Goyal, Mike Hibler, David Johnson, Tim Stack, Leigh Stoller, and Kirk Webb; to Dave Andersen and Nick Feamster for the Datapository, to Dave for helpful discussion, to Ken Yocom and the reviewers for their useful comments, and to NSF for its support under grants CNS–0335296, CNS–0205702, and CNS–0338785.

References

[1] J. Albrecht, C. Tuttle, A. C. Snoeren, and A. Vahdat. PlanetLab Application Management Using Plush. *ACM SIGOPS Operating Systems Review*, 40(1):33–40, Jan. 2006.

[2] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *18th SOSP*, pages 131–145, Mar. 2001.

[3] D. G. Andersen and N. Feamster. Challenges and Opportunities in Internet Data Mining. Technical Report CMU–PDL–06–102, Carnegie Mellon University Parallel Data Laboratory, Jan. 2006. http://www.datapository.net/.

[4] L. Brakmo, S. O'Malley, , and L. Peterson. TCP Vegas: New techniques for congestion detection and avoidance. In *SIGCOMM 1994*, pages 24–35, Aug. 1994.

[5] M. Coates, A. O. Hero III, R. Nowak, and B. Yu. Internet Tomography. *IEEE Signal Processing Magazine*, 19(3):47–65, May 2002.

[6] S. Floyd and E. Kohler. Internet Research Needs Better Models. In *First Workshop on Hot Topics in Networks*, Oct. 2002.

[7] S. Floyd and V. Paxson. Difficulties in Simulating the Internet. *IEEE/ACM Transactions on Networking*, 9(4):392–403, Aug. 2001.

[8] P. Francis, S. Jamin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. IDMaps: A Global Internet Host Distance Estimation Service. *IEEE/ACM Transactions on Networking*, 9(5):525–540, Oct. 2001.

[9] S.-J. Lee et al. Measuring Bandwidth Between PlanetLab Nodes. In *First Passive and Active Measurement Workshop*, Mar. 2005.

[10] X. Liu and A. Chien. Realistic Large-Scale Online Network Simulation. In *2004 ACM/IEEE Conference on Supercomputing*, Nov. 2004.

[11] H. V. Madhyastha et al. iPlane: An Information Plane for Distributed Services. In *Seventh OSDI*, Nov. 2006.

[12] A. Nakao, L. Peterson, and A. Bavier. A Routing Underlay for Overlay Networks. In *SIGCOMM 2003*, pages 11–18, Aug. 2003.

[13] T. S. E. Ng and H. Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. In *IEEE/ACM Transcactions on Networking*, Oct. 2001.

[14] D. Oppenheimer, B. Chun, D. Patterson, A. C. Snoeren, and A. Vahdat. Service Placement in a Shared Wide-Area Platform. In *2006 USENIX Annual Technical Conf.*, pages 273–288, May–June 2006.

[15] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *First Workshop on Hot Topics in Networks*, Oct. 2002.

[16] L. Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *SIGCOMM CCR*, 27(1):31–41, Jan. 1997.

[17] J. Sommers, P. Barford, N. Duffield, and A. Ron. Improving Accuracy in End-to-end Packet Loss Measurement. In *SIGCOMM 2005*, pages 157–168, Aug. 2005.

[18] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP Topologies with Rocketfuel. In *SIGCOMM 2002*, pages 133–145, Aug. 2002.

[19] N. Spring, L. Peterson, V. Pai, and A. Bavier. Using PlanetLab for Network Research: Myths, Realities, and Best Practices. *ACM SIGOPS Operating Systems Review*, 40(1):17–24, Jan. 2006.

[20] N. Spring, D. Wetherall, and T. Anderson. Reverse-engineering the Internet. In *Second Workshop on Hot Topics in Networks*, Nov. 2003.

[21] W. A. Taylor. Change-Point Analysis: A Powerful New Tool for Detecting Changes. http://www.variation.com/cpa/tech/changepoint.-html, Feb. 2000.

[22] A. Vahdat et al. Scalability and Accuracy in a Large-Scale Network Emulator. In *Fifth OSDI*, pages 271–284, Dec. 2002.

[23] A. Vahdat, L. Peterson, and T. Anderson. Public statements at PlanetLab workshops, 2004–2005.

[24] K. Webb et al. Implementing the Emulab-PlanetLab Portal: Experience and Lessons Learned. In *First Workshop on Real, Large Distributed Systems*. USENIX Association, Dec. 2004.

[25] B. White et al. An Integrated Experimental Environment for Distributed Systems and Networks. In *Fifth OSDI*, pages 255–270, Dec. 2002.

[26] K. Xu, Z.-L. Zhang, and S. Bhattacharyya. Profiling Internet Backbone Traffic: Behavior Models and Applications. In *SIGCOMM 2005*, pages 169–180, Aug. 2005.

[27] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang. PlanetSeer: Internet Path Failure Monitoring and Characterization in Wide-Area Services. In *Sixth OSDI*, pages 167–182, Dec. 2004.

[28] Y. Zhang, N. Du, V. Paxson, and S. Shenker. On the Constancy of Internet Path Properties. In *SIGCOMM Internet Measurement Workshop*, pages 197–211, Nov. 2001.

Interconnection Discrimination: A Two-Sided Markets Perspective

P. Faratin†, T. Wilkening††

† Computer Science and AI Lab, †† Department of Economics, M.I.T

Abstract

We first motivate a methodological approach to network economics research using tools from Industrial Organization (IO). We then demonstrate the utility of such models in light of an increasingly important problem of interconnection failures that includes network neutrality problem. In particular, we focus on a subset of the problem by focusing on direct interconnection failures when ISPs price discriminate between content providers and content users. We informally show how viewing the problem as a Two-Sided Market can lead to a discrimination rule which highlights the structural/distributional nature of the interconnection problem. Structural consideration has not only policy and antitrust consequences but can also be highly informative to engineering choices.

1. Introduction

The networking research community has used game theory and mechanism design tools to explain and design against various strategic behavior of entities in forwarding, routing, congestion control, Ad Hoc networking, caching and P2P problems. The methodology of the majority of these applications has been to focus on short-term strategic behavior of individual ASs or edge nodes in a large network. However, although useful this methodology does not capture the important *market* variables that determine these short-term behaviors. AS's routing and capacity expansion decisions for example are based on long-term exogenous and endogenous factors such as (threat of) competition and regulation, investment "holdups" and general business risks and uncertainty. The high level goal of this paper is to introduce the community to another branch of economics called Industrial Organization (IO) which takes the *firm* as the basic unit of economic analysis, whose strategic actions can have large and sometimes undesirable (anti-competitive) impact on the network on both the short and long term.

We believe IO is the right tool to consider for networking research for both methodological and design reasons. Methodologically IO models take the firm as the basic unit of analysis and typically include primitives such as the structure of the market (monopoly, duopoly, oligopoly), endogenous and exogenous production and cost structures, nature of demand (elasticities), nature of the goods (complements, substitutes, storability) and the actions available to the agents (choice of mechanism designer, price versus quantity selection, regulatory environment). These endogenous variables have first-order effects on the behavior of an ISP. Current network economics models seriously run the risk of "throwing away the baby out with the bath water" by abstracting away these important variables. Li et.al [Li04] demonstrated an equivalent point that statistical (power-law distribution) models of a network topology do not in fact have much explanatory power because they often lack finer micro-details of engineering constraints that better explain and predict network structure. In a similar manner the micro-details of the market critically matter in forming incentives and network economics models must address these details in the design phase and resist assuming separability of economic and technical design by "over the wall" design.

The second rationale for adopting IO models is constructive. IO models may provide us with information that is useful in thinking about operational as well as architecture and protocol design choices. Internet researchers often make certain structural and behavioral assumptions when designing for the Internet. AS level source routing for example assumes multiple layers of hierarchy. Yet, the Internet is actually becoming increasingly flat as access networks grow and interconnect with backbones through a single hop transit. Why is the Internet becoming flatter? Could a theory explain and predict such structural changes as flattening of the hierarchical peering/transit relationships? IO models of other industries shows why firms may at times have very strong economic incentives to vertically integrate with upstream providers and scale so as to recover high fixed costs, lower variable transit costs and reduce hazards in investments [TIR88]. Additionally, could a theory also explain the long-run *behavioral* rationales of ISPs such as their discrimination strategies (through economic instruments such as vertical integration or foreclosure, exclusive dealing and bundling using pricing and technology)? Such a theory can be immensely valuable for evaluating not only current but potential future (v. GENI and FIND) operational and engineering choices. IO models have indeed been used for analysis by practitioners in policy, legal and anti-trust cases to frame problems such as structural separation of transport from information services, predatory pricing and anti-competitive behaviors in not just Internet but many other industries. Can IO models also serve to inform operators and engineers who operate and design for the Internet and who maybe able to intelligently *shape* the outcomes through design of protocols that change the rules and strategy of the game that can be played by ISPs? Note,

such a design-evaluate constructive methodology is a departure from the specify-design methodology of majority of current network economics models (viz. mechanism design). In the latter case engineering constraints are often added *after* the economic properties (incentive compatibility, efficiency, individual rationality, budget balance) are satisfied. For example, a smart market mechanism has many desirable economical properties but is technologically infeasible in the current Internet architecture (every packet has to carry a bid). A design-evaluate (closed-loop) construction methodology on the other hand lets engineers *first* design for technical constraints and only then evaluates the economic effects (by considering the game and the equilibria that is induced by a technical design).

We believe these methodological and constructive goals constitute a novel and interesting long-term networking research agenda. In this paper we demonstrate the usefulness of IO models and methodology by focusing on the concrete problem of discriminatory *price* behavior of ISPs toward content providers under two different structural setups (single and multi-homed). This problem is an instance of a more general interconnection breakdown that arises due to discrimination by ISPs and is a subset of problems in the network neutrality debate. The central contribution of this paper is to show how a new body of IO literature, called Two-Sided Markets (TSM), is providing a testable causal model of discrimination by showing that discrimination may in fact have a rational basis when demand interdependencies *across* markets, multi-homing and non-linear tariffs are taken into considerations. The demonstration of full richness and applicability of these models in the Internet is beyond the scope of this paper. Therefore the goal of this paper is to only informally show how such models can address discrimination concerns and can even be useful when forming operational and engineering expectations over interconnections.

2. The Problem: Interconnection Discrimination

Failures to coordinate and implement differentiated End-to-End QoS, increased competition and lack of innovations are increasingly making the Internet transport service a commodity service. This together with decreasing marginal uptake of broadband services is lowering the margins of transport providers whose incentive to (further) discriminate is increasing. Current network neutrality debates are centered round a number of such discrimination behaviors ranging from non-priced based discrimination practices such as structural (when ISP vertically integrates into content and/or applications), bit and packet level discriminations. In this paper we focus on neutrality problems involving interconnection breakdowns on the *access links* when Access Providers (APs) price discriminate, a problem that is increasing in frequency, most notably when BellSouth threatened Google with higher charges in 2006. We restrict ourselves to this class of discrimination problem by focusing on two markets, content consumers and providers, which seek *direct* "on-net" interconnection through the AP transport services. ISPs, specially smaller scale ones who cannot enjoy scale economies, have a strong incentive to serve content "on-net" so as not only reduce usage-based transit costs but also increase revenues through payments by content providers. Entry by third-party content distribution overlay networks such as Akamai is a strong market signal of the cost-minimization importance of this class of interconnections (see Clark et.al. for an exposition, [CLA05]). We therefore do not consider ISP interconnections, when requested content must be served "off-net" in an end-to-end manner. The nature and extent of economic interactions that can occur between on-net and off-net interconnections is beyond the scope of this paper, and the interested reader is referred to Laffont et.al for an in-depth IO model of the economic (efficiency and welfare) role of interconnection charges on perceived costs of ISPs with a mixture of on and off net traffic [LAF03]. Specifically, we make the following simplifying assumptions. Firstly, APs compete for the consumers *before* the user commits to a single AP. But after the consumer has committed (is single homed) that AP is a monopolist, at least as long as there exists substantial switching costs. Secondly, the market share of the consumers is constant. Third, we assume traffic is "on-net" and there is no other indirect (peering) path to the consumers other than through the monopolist; the AP is a "competitive bottleneck" [ARM05]. Fourthly, because of previous assumption and because the content provider wants to be in contact with the widest population of consumers, content providers multi-home to a number of APs.

The network neutrality debate in this restricted setting often centers around tariff uniformity which states that the interconnection settlement tariffs should be identical for the best-effort class of service, because in absence of QoS, transport is an undifferentiated good and so there is no economic basis for an ISP to price transit contracts differently between, say, Google and a small content provider. In other words a network should offer uniform tariffs and not discriminate against the type/label of the customer (i.e. no third-degree price discrimination [TIR88]). Opponents of uniformity claim this is an over-simplification because what constitutes discrimination is in fact a difficult problem from an economic perspective (indeed, competitive equilibrium severely restricts the set of potential discriminations because goods sold in different times at different state of nature are in fact *different* goods). For example, is an ISP price discriminating when it offers volume discount transit contracts? It is offering the same service independent of

volume. The answer is not clear and depends on what cost perspective is adopted. If aggregate traffic is below capacity then such behavior can be deemed undesirable from a usage cost perspective. However, under an opportunity cost perspective discrimination is rational, since traffic is being priced according to the use the ISP could have otherwise put the consumed bandwidth to. Equivalently, whether interconnection prices are discriminatory is equally dependent on the perspective adopted. Below we will show that adopting a perspective that content users and providers are two non-separable and interdependent markets can result in an interconnection discrimination rule that can be qualitatively very different than if a single-sided market perspective was adopted. A prescriptive result from the theory useful for construction by an engineer is that in such (two-sided) markets multi-homing should not be considered only from a resilience perspective. Multi-homing in fact has major economic impact on the prices and ultimately on connectivity of Internet.

3. Two-Sided Markets: Evidence

Classic IO and multi-product literatures show that firms have an incentive to discriminate through both prices (first, second and third degree pricing) and non-price (vertical foreclosure, bundling, exclusive contracts) instruments so as to maximize the amount of surplus they can extract from consumers [TIR88]. The goal in these discriminatory mechanisms is to *capture* (rather than create) maximum surplus ("rent seeking" behavior). Use of coupons, group discounts and bundling are some examples of commonly practiced discrimination strategies in many markets which is also prevalent in the Internet retail sector. For example, retail peak-rate pricing tariff is (second-degree) price discrimination, where the (monopolist, at least during the contract period) operator presents a menu of tariffs that support different peak-rates and customers choose a tariff, based on their private preferences (or "type"). This type of "one-sided" discrimination, where the firm contracts directly with only the consumers in one market, has been well understood and is commonly practiced in the communication as well as many other industries.

However, the central intuition of this paper is that such one-sided discrimination arguments are inherently misguided in interconnection debates because interconnection requires coordination of value-flows *across* a number of markets, not just one. Data and communication services require the coordination of multiple markets including content providers and content users and callers and receivers respectively. The key feature of these interconnection problems is that an ISP has to solve the "chicken-and-egg" problem of bringing "onboard" both content users and content providers. Furthermore, these problems are continual and dynamic since customers are poached by or switch to other providers. For instance, Akamai can be seen to be capturing content providers who were previously served by backbones and who have a high willingness to pay for high quality low latency transportation, a service that is currently unavailable in an end-to-end manner and involving transit across peering points which do not permit marginal payments for higher quality service (peering points are "money insulators", [CLA05]). The chicken-and-egg problem is in fact a prominent problem in a diverse set of "platform" industries ranging from operating systems to search engines, nightclubs, computer games console, auctions, exchanges, credit cards, money and real estate agencies, to name a few, where there is strong cross-market externality/spillover and the volume of demand in one market is highly dependent on volume of demand in the other market [ROC05,EVA03]. Informally, nightclubs need to ensure both men and women are "onboard" for the "platform" to be profitable. Operating systems can be seen as a platform where application developers (one market) develop application and consumers (the other market) pay a transaction independent price for use of the platform and associated applications. Auctions houses such as eBay or Christies, facilitate exchange between buyers and sellers through lowering transaction and information costs. Credit cards such as Visa connect buyers with merchants. Even intangible services such as money can be viewed as a platform that through mandating a standard facilitates exchange. Similarly, an ISP can be viewed as a platform in a two-sided market, interconnecting content provider and content user markets (see [ROC05] for a formal definition of a TSM).

The critical feature of all these diverse examples of platforms is indirect value flow (a.k.a. *indirect network externalities* [ECO92]); that one/both sides of the market benefit from increasing adoption and/or consumption of the other side. Advertizers are attracted more to Google than other search engine because of scale of Google's searchers. More buyers transact over Ebay than other competing auction platforms because there are more sellers on eBay, who in turn are attracted to the platform because of buyer market size. Men value nightclub venues with more women. Developers are attracted to Microsoft platform because more consumers have adopted Windows. ISPs with many "eyeballs" want to interconnect with Cogent who hosts most of the porn on the Internet. The key problem for platforms, such as ISPs, in such externalities is how to maximize profits by recognizing, managing and harnessing these cross-market value flows (or externalities) that lead to increased adoption and/or transaction volumes. Bilateral contracting and transactions between the platform and each side of the market independently of the other side (such as peak-rate pricing) can in fact lead to less than optimal profits for the platform [ROC05].

In practice pricing in two-sided markets often takes the form where one side is subsidized, possibly even below marginal cost to the platform. Such "loss leader" markets are very prevalent. For example, the loss leader on Google's search platform (as well as many other media platforms such as magazines, TV, radio, etc) is the searcher/viewer market. The platform charges advertisers instead. Real estate agencies only charge the seller market a fixed membership charge, providing services to the buyers for free. Credit cards such as Visa charge only a yearly membership charge (which is sometimes even below marginal costs, that is a gift) to the consumer and a transaction charge to the merchants in the form of merchant discount. Nightclubs do not charge women (who in fact may even be subsidized with free drinks), whereas men can be charged both entrance and usage fee. Auctions mostly charge sellers and not buyers a transaction-independent price. Interestingly most operating systems and computer games adopt the reverse strategy, charging the consumers a fixed price (through licensing) while subsidizing the developer markets below cost. Pricing discrimination is apparent in the communication and Internet sectors too. Akamai charges content providers and not access networks for its services (unlike Inktomi who instead charged ISPs and not content providers, and who finally exited the market). ISPs often practice "double billing", charging both unaffiliated content providers and content users. EU mobile operators charge callers and not receivers. Some networks that have large number of "eyeballs" on their network charge below cost transit to "popular" content providers relative to comparable content providers who have equal content volumes but fewer eyeballs.

4. Two-Sided Markets: Theory

In most industries such discriminatory pricing practices are often interpreted as predatory by lawyers and antitrust authorities, because the incumbent charges below cost to capture the market and force exit of competitors, after which can behave as a monopolist and extract more of consumer surplus. Areeda and Turner, two lawyers, laid the foundations of much of current thinking about predatory pricing used by anti-trust authorities. But Areeda and Turner's discrimination rule (if prices are below marginal cost then the firm is behaving anti-competitively), and even market definition, is erroneous if the chicken-and-egg and cross-externalities problem facing the firm is admitted into considerations. The explanatory power of TSM theory is that it not only describes the common structure of divergent industries but more importantly defines a relevant and rational basis for platform discrimination between markets. Recall that price discrimination is a concern in network neutrality because of increased likelihood of interconnection breakdowns. But in a TSM one side is subsidized, possibly even below marginal cost to the platform, to induce growth in that market, with the expectation that the growth in one-market due to lower prices will, through a positive feedback loop, induce positive growth, increased prices and supra-profits in the other market [EVA03,PAR05,ROC05]. In fact, even a monopolist will have an incentive to cross-subsidize markets. Therefore in TSMs the relevant measure is not the level of prices (as in traditional methodology) but rather the *structure* of prices. Distributional considerations, usually concern of anti-trust authorities, were also central to structural reforms of the Telecommunication regulation, where rate of return regulations allowed the incumbent operators to adjust prices on different lines of business according to the elasticities of demand for each product, so as to recover fixed costs of network investment and universal services [NUE05]. Both TSMs and line of business regulation therefore are concerned about structural problems, resulting in the famous Ramsey program [LAF02]. However, the difference is that the concern in telecommunication domain was over pricing structure of multiple products (line of business) whereas in TSM the concern extends *across* multiple markets that have cross externalities. It is in this sense that theory of TSMs unifies multi-product (first formalized under the work of W. Baumol) and network externalities literatures. The exact nature and magnitude of such cross-subsidies is generally dependent on not only the sensitivity of demand in each market to prices (price elasticity of demand) but also: i) the degree of cross-market elasticities, ii) extent of multi-homing and iii) the degree of membership and usage externalities [ARM05,ROC05]. Below we will briefly cover the first two dependencies.

Figure 1 shows the geometry of the pricing problem of a monopolist ISP who intermediates two (single-homed) markets i and j. Let the i and j markets be the content user and producer markets facing usage prices p_i and p_j from the ISP respectively. The problem of the ISP is to determine the structure of profit maximizing prices. Let q_k denote the total consumption of network transport services in market $k \in \{i,j\}$. One potential additive demand function is $q_i = D_i(P_i) + e_{ji}D_j(P_j)$, where $D_i(P_i)$ is the "native" demand in the i market at price P_i, and the additive term is the effect of the consumption in the other market (see [PAR05] for nonlinear demand). The constant $e_{ji} = \partial q_i / \partial q_j$, measures the marginal change in consumption in market i with a marginal increase of consumption in the j market and represents the externality/spill-over effect market j consumption has on market i demand. Figure 1, shows the *benchmark* pricing structure case where there is no cross-market effects, $e_{ji} = e_{ij} = 0$. The solid lines represent the pricing reaction curves $p_i(p_j)$ and $p_j(p_i)$, representing the optimal prices in the i market given prices in the j market and, conversely,

the optimal prices in the j market given prices in the i market respectively. Specifically, $p_i(p_j)$ is computed as the solution to the following maximization problem with p_j fixed:

$$p_i^*(p_j) = \arg\max_{p_i} (p_i q_i) + (p_j q_j).$$

When there is no cross-market effects, $e_{ji} = e_{ij} = 0$, the equilibrium set of prices in each market lies on the 45° line, corresponding to classic monopoly prices. That is, when markets are independent then both markets are priced positively (the level of which is captured by the Lerner index and regulated by the degree of elasticity of demand in each market, [TIR88]).

As mentioned above the majority of ISPs charge positive prices for both content requestors and servers ("double billing", quadrant I). However, as also mentioned above, there is considerable heterogeneity in pricing in the Internet where ISPs also charge considerably less (even below cost) to content providers whose content is much in demand. Figure 2 shows how the pricing structure can diverge from the benchmark independent markets when the cross-market effect from market i (content users/eyeballs) is constant, and cross-market effect from market j to market i increases [PAR05]. That is, as e_{ji} increases (the demand of content users increases as demand for transport by content provider increases) then prices to the content providers decrease, to the point (e_{ji} = 11/10) that the content providers may in fact be subsidized by the platform (because users value content).

The above reasoning is a plausible model of observed pricing structure between "valuable" content providers and users. However, pricing structure may also be skewed and perceived to be discriminatory when one or both sides of the market are multi-homed to competing platforms [ARM05,ROC05, HER06]. Multi-homing is typically viewed from an operational and engineering perspective as a resilience mechanism. Routing overlays can also be seen to be multi-homed across both the layer 3 and the overlay so to choose the best quality routing path to solve the triangular inequality problem. However, there are other endogenous economic reasons for multi-homing (from both the ISP and the node perspectives) other than resilience. Firstly, competition among ISPs must be accounted for in the reasoning when multi-homing. ISPs are in strong competition with one another, differentiating their services so as to increase prices and maximize "rents" that can be extracted. Callers have a choice of mobile or fixed line telephony. Cable competes with DSL along number of service dimensions including speed, customer support, available content, etc. At times firms have an incentive to be exclusive with one side of the market in order to build market share. Incumbent Real Media, for example, tied content providers to propriety Real content format for its players through exclusive contracts so as to compete with the entrant Windows media player, forcing consumers to "multi-home" to both Media and Real players. In general increasing platform differentiation forces one or both side of the market to have an economic incentive to multi-home. However, the TSM literature is beginning to show that the platform in fact would prefer unilateral multi-homing on only one side of the market. When *both* sides multi-home then the pricing structure is closer to cost, lowering potential monopoly profits. For instance, in credit card industry (which incidentally shares many interconnection features as the Internet – see [ROC05]), credit card companies court

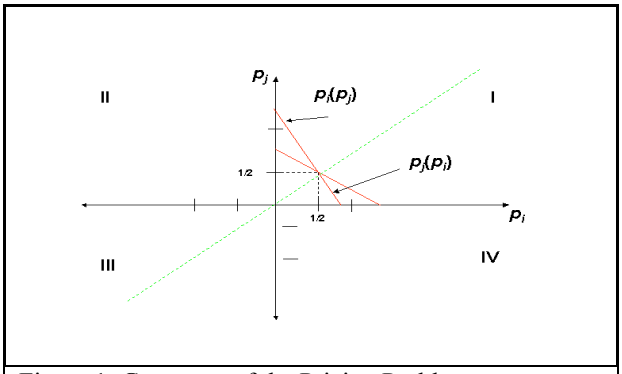


Figure 1: Geometry of the Pricing Problem

multi-homed merchants (merchants who accept a number of cards) much more aggressively (through

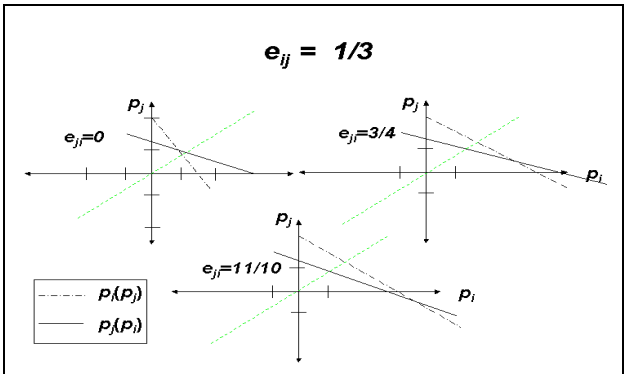


Figure 2: Pricing with Indirect Externalities

lower merchant charges) when cardholders are also multi-homed (carry multiple cards). The reader is referred to [HER06] for an in-depth analysis when both sides of the market multi-home (a literature we believe can give insights into economics of source-routing). However, when only one side of the market is multi-homed then the pricing structure of the ISP can be significantly above cost to the multi-homed side. To see this we continue to restrict ourselves to the case where the platform is a monopolist on the content users side (who are single homed at least for the duration of

contract), but now the content providers are multi-homed, better representing the problem between Google and BellSouth. Armstrong shows the equilibrium price structure in an equivalent market setup between mobile users (content users) and fixed line users (content producers) who wish to call the mobile users [ARM05]. He shows that the "competitive bottleneck's" equilibrium price structure is to set low (in his model subscription) charges to single-homed side and high termination charges to the multi-homed side. The model also predicts that the high prices made on the higher termination charges to the multi-homed side are passed onto the single-homed subscribers in the form of subsidized services.

Multi-homing can therefore dramatically influence the pricing structure across markets. These consequences can be either positive (as in subsidies to single-homed users) or negative (as in above cost markup charges), *if we adopt a single sided perspective*. One prescription of the theory to an operator is to make sure multi or single homing decisions are not made myopically based solely on resilience criterion, but rather consider the connectivity and the cross-market benefits. Carefully consider the benefits of multi-homing expecting higher costs, if the other side is single-homed. A stronger prescription from the theory would be to engineer protocols and architect into the network that if multi-homing is a choice then it is enforced by default. Such configurations will result in lowering price distortion incentives by the ISP. However care should be taken because under a TSM view there *can* be flows of cross-subsidies whose benefits maybe on a longer time scale. Higher costs to Google, for example, today means lower prices for users which in turn may result in increase in demand for the ISP services which will in turn result in more eyeballs for Google's advertisers tomorrow.

5. Conclusion

We have presented an Industrial Organization framework to model on-net interconnection breakdowns that can result from price discrimination and showed that under some circumstances discrimination is in fact rational and both businesses *and* engineers would be prudent to consider these cross-market effects; interconnections occurs not only at layers 2 & 3, but also at the level of markets. The long-term goal of our research agenda is to use IO economic theory to describe economics of value flow in End-to-End off-net interconnection and "coopetition" behavior of ISPs, content providers and content overlay networks. The credit card network, where competing banks form a cooperative to allow customers to seamlessly transact with any merchant, is an interesting industry that shares many of 'coopetition" and "routing-money" problems as the Internet. Models developed there may in fact be useful. We believe such a long-term research agenda is methodologically closer to networking research community goals who view engineering constraints as first-order constraints, followed by other economic and social constraints. Considering, rather than satisfying, economic constraints early on in the design stage is important if innovations are to enter the market. IO, as a maturing discipline, is beginning to provide such analysis tools.

Bibliography

[ARM05] M. Armstrong (2005): *Competition in Two-Sided Markets*, forthcoming, Rand Journal of Economics, forthcoming

[CLA05] David Clark, William Lehr, P. Faratin, S. Bauer and J. Wroclawski (2005): *The Growth of Internet Overlay Networks: Implications for Architecture, Industry Structure and Policy* In the proceedings of Telecommunications Policy Research Conference (TPRC-05), Washington, DC. 2005.

[ECO92] N. Economides and Salop (1992): *Competition and Integration Among Complements, and Network Market Structure*, in Journal of Industrial Economics, vol. XL, No. 1, March 1992, pp. 105-123

[EVA03] D. Evans (2003) *The Antitrust Economics of Two-Sided Markets*. Yale Journal on Regulation: 2003, 20(2), pp: 325—381

[HER06] B. Hermalin and M. Katz (2006), *Your network or mine? The economics of routing rules*, RAND Journal of Economics, forthcoming

[LAF02] J.J. Laffont and J. Tirole (2002): *Competition in Telecommunication*, MIT Press, Cambridge, MA, US, 2002.

[LAF03] J.J Laffont, S. Marcus, P.Rey and J. Tirole (2003): *Internet Interconnection and the Off-Net Pricing Principle*, RAND Journal of Economics, (34), 2, 2003, pp. 370-390.

[LI04] L. Li, D. Alderson, W.Willinger and J. Doyle (2004): *A First-Principles Approach to Understanding the Internet's Router-Level Topology*, SIGCOMM 2004, Portland, Oregon, USA, pp. 3-15.

[NUE05] H. E. Nuechterlein and P.J. Weiser (2005) *Digital Crossroads: American Telecommunications Policy in the Internet Age*, MIT Press, Cambridge, MA, US, 2005

[PAR05] G.G.Parker and M. Van Alstyne (2005): *Two-Sided Network Effects: A Theory of Information Product Design*, Management Science, (51), 10, 2005, pp. 1494-1504.

[ROC05] J. Rochet and J. Tirole (2005): *Two sided Markets*: A Progress Report, forthcoming, Rand Journal of Economics. (http://idei.fr/vitae.php?i=51)

[TIR88] J. Tirole (1988): *Industrial Organization*, MIT Press, Cambridge, MA, US

Achieving Good End-to-End Service Using Bill-Pay

Cristian Estan Aditya Akella Suman Banerjee
University of Wisconsin-Madison

1 Introduction

Over the past couple of decades, the Internet has rapidly evolved from a collaborative social experiment to an agglomerate of competing commercial providers. This shift has helped maintain growth and has turned the Internet into a vast economic force, but it has also introduced some serious problems. A particularly bad problem is the inability of end-users to obtain the desired levels of performance for their transfers.

Today, an organization can set up a contract with its ISP to ensure that the ISP offers good service to its traffic. But typical transfers in the Internet traverse multiple ISPs and it is clearly infeasible for the organization to have contracts with all of them. It is possible for neighboring ISPs to enter into contracts that require them to offer good performance to each other's "premium" traffic. However, such contracts are extremely rare and, even when used, cannot guarantee good end-to-end service to user transfers.

Our thesis in this paper is that we can support good end-to-end service to user transfers by extending the current model of binding bi-lateral contracts between neighboring entities (e.g. customers and providers or peering partners) with simple mechanisms that produce *tacit incentives for remote ISPs*. Our use of the phrase "good end-to-end service" is intentional: our goal is not to offer "end-to-end QoS" with strict performance guarantees, but rather to provide end users the *flexibility to improve* the performance experienced by their transfers, as and when desired.

Our proposal builds on two main end-user based mechanisms that generate the tacit incentives.

- **The carrot:** We propose that the end-user include in-band payments with their data. Each ISP then retains a portion of the payment, commensurate with the transit performance it offers.
- **The stick:** We propose that end-users be able to influence what path their traffic uses and thus bypass remote ISPs with unjustifiably bad service and/or those requiring unjustifiably high payments.

Our proposal, *Bill-Pay* (*Bil*ateral *l*ocal nano*Pay*ments), enables end-users to apply these mechanisms in a fine-grained manner by adding, to every individual packet: (1) a small payment which we call "nanopayment", and (2) information indicating the user's preferred path and service levels. In its basic form, *Bill-Pay* users pay according to their network usage, but *Bill-Pay* can support "flat fee" pricing for end users as well. We argue that *Bill-Pay* enables good service to end-user transfers and allows more effective protection against DDoS floods and spam.

1.1 Overview of *Bill-Pay*

We illustrate the functioning of *Bill-Pay* using the example in Figure 1. End-host *A* wishes to transfer data to end-host *B*. ISP *Y* along the path experiences congestion that reduces the throughput of the transfer below that desired by *A*.

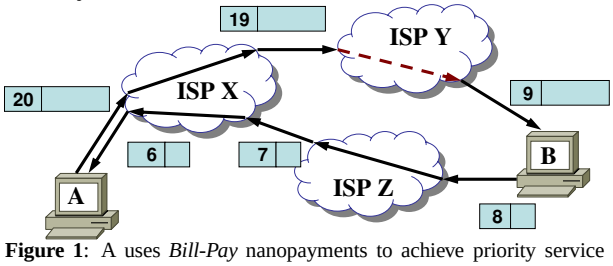


Figure 1: A uses *Bill-Pay* nanopayments to achieve priority service through congested ISP Y.

All networks and users in this example have local bilateral *Bill-Pay* agreements with their neighbors (*A* has an agreement with ISP *X*, *X* and *Y* have an agreement, etc.). To improve throughput in the face of congestion in ISP *Y*, *A* adds a nanopayment of 20 nanodollars to the data packet it sends to *B*. Since ISP *X* is not congested it leaves most of the nanopayment in the packet as it forwards it to ISP *Y*. ISP *Y* retains 10 nanodollars and gives the packet preferential treatment. *B* returns most of the remaining nanopayment in the acknowledgment packet it sends through ISP *Z*.

We note that *A* owes ISP *X* 14 nanodollars for this particular pair of packets: while *A* sent out 20 nanodollars, it received 6 nanodollars in the acknowledgment. In return, *A*'s packet received good service despite the congestion in ISP *Y*. These nanopayment balances are converted to actual payments at the end of the billing cycle. We note that the profit made by an intermediate ISP is related to the number of packets it carries, and the level of service it offers. For example, *X* makes 2 nanodollars for two packets and *Z* makes 1 nanodollar for the one packet (both *X* and *Z* offer "regular" service) while *Y* makes 10 nanodollars for offering premium service to a single packet.

The viability of such an architecture depends on four important questions.

- **How can we ensure that ISPs provide improved service at fair prices?** In Section 2.1 we discuss the incentives ISPs have to limit the amount of nanopayment they retain and to provide a commensurate service quality. We also discuss various mechanisms which ISPs can use to determine how much payment to retain.
- **How can *Bill-Pay* end-users avoid expensive and congested paths?** In Section 2.2 we describe a mechanism that allows the senders to influence the trajectory of packets through the network at a granularity slightly finer than AS-level paths, with ISPs retaining control over the level of detail exposed to end-users.
- **How can the end-user ascertain how large a payment a transfer is worth?** In Section 2.3 we discuss the feasibility of a digital secretary that learns the user's preferences.
- **How can the payments be secured from malicious hackers who take control of an end-host?** In Section 2.4 we discuss mechanisms that ensure that even if the end-host is hijacked, no significant payments can be leaked without the consent of the user.

After discussing each of these issues in turn, we examine how our architecture facilitates solutions to various important problems (Section 3), how it compares to prior related proposals (Section 4), and finally conclude with a discussion of future work (Section 5). The technical report version of this paper [1] also contains a discussion of how *Bill-Pay* can interoperate with existing technologies such as diffserv and how it can be incrementally deployed.

2 Detailed discussion of *Bill-Pay*

The basic *Bill-Pay* contract is very simple and very easy to enforce: the upstream organization has to pay the downstream an amount of money equal to the total of the nanopayments in the packets it sent, and the downstream organization has no contractual obligation. Since the downstream organization has an *economic incentive* to provide good service to the packet, contractual obligations are not needed. More complex contracts linking payment to performance metrics such as loss rate and jitter clearly provide a stronger incentive for the downstream ISP to offer good service to the selected packets, but they require trustworthy measurements of the degree of compliance and the sender needs contracts with all ISPs on the path. In contrast, with *Bill-Pay* the incentives "carry over" along an end-to-end path, even without an explicit contract.

In its simplest form, *Bill-Pay* enables unidirectional nanopayments: the sender is the ultimate upstream and the origin of the nanopayment and all organizations on the path of the packet can retain a portion of the nanopayment. However, in a web browsing scenario (and in many other settings), it is common that the receiver of the packets is the one willing to pay to improve QoS (irrespective of whether the congestion is on the path to, or from, the sender). *Bill-Pay* can easily handle such a scenario with the cooperation of the server: the client sends nanopayments to the server throughout the lifetime of the TCP connection, and the server puts the remaining amount in the packets carrying content. To simplify presentation, in the rest of this paper we assume that the source is the one paying for the network traffic.

2.1 ISP Behavior and incentives

Bill-Pay can deliver benefits to end users only if most ISPs provide an appropriate level of service to packets and retain a reasonably low amount from the nanopayments. Later in this section we discuss the incentive structure which will motivate ISPs to adopt such acceptable behaviors. We start by detailing a central aspect of ISP behavior: the method used to decide the amount to retain from the nanopayment in the packet, which we call a "toll". We argue that at least two toll mechanisms are needed: congestion-based and fixed [1].

Congestion tolls on packets are to be set dynamically based on the level of congestion on links being traversed by the packets. All packets with nanopayments above the congestion toll pass and the congestion toll is deducted. The packets with nanopayments smaller than the congestion toll get dropped with probability proportional to the difference between the toll and the nanopayment. There are two important decisions to be taken at such congested links under this toll model — (i) congestion pricing: the amount of congestion toll to be retained from the packets, and (ii) congestion scheduling: the order in which packets are processed at the congested links. Both of these decisions depend on the level of congestion and the amount of nanopayments included in the packets. We note that the congestion toll can also be used to signal congestion to all senders, analogous to the way packet losses are used by TCP today.

A significant amount of past work has addressed the former issue of congestion pricing – but only in the case of a single ISP (discussed further in the related work). Congestion scheduling, in contrast, is a relatively unexplored area. We hope to address both of these challenging issues in future work.

While congestion tolls are an useful construct, this mechanism alone is not sufficient to guarantee reasonable ISP behavior, because it gives ISPs an incentive to create "fake congestion" in their networks to collect

[1] Other types of tolls such as proportional tolls (a percentage of the nanopayment) are also possible, but we do not discuss them here.

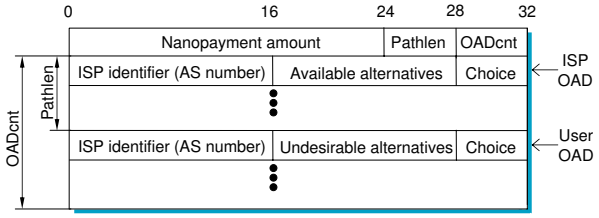


Figure 2: The *Bill-Pay* header structure

more money from packets. This motivates the case for fixed tolls, described next.

Fixed tolls are independent of the level of congestion in the network. They are a suitable mechanism to recuperate the sunken costs of running the network. Based on today's prices, these fixed tolls can be on the order of nanodollars per packet and picodollars per byte, but the amount depends on technology and on the strength of the incentives for keeping tolls low.

If ISPs also collect fixed tolls, in addition to congestion tolls, an ISP artificially inflating congestion tolls faces a loss of fixed toll revenue due to traffic that shifts to other ISPs and this acts as a deterrent for fake congestion tolls.

2.1.1 ISP incentives for acceptable behavior

It may appear that a greedy ISP receiving packets with *Bill-Pay* nanopayments could keep the money (i.e., charge a very high toll) and drop the packet. Even without nanopayments, dropping packets is cheaper than carrying them. But just as we do not see this type of nearsighted greedy behavior with today's ISPs, we expect that ISPs with *Bill-Pay* contracts will not behave in this negative fashion either. The core motivation in both cases is the *promise of future payments*.

Competition is the most important incentive for ISPs. If there is enough path diversity between the sender and the receiver and one ISP imposes unreasonable tolls, the sender can shift subsequent traffic to a different path. While single-homed users must send all their packets through the single ISP they connect to, the threat of them switching to a different ISP provides an incentive for keeping the tolls low. Note that it is not an economic or technological requirement that high speed Internet access be a choke point with high tolls. In the Utopia project [10], for example, access links are managed by a community-owned organization and the users can easily choose between many ISPs who can offer service through these access links.

Legislation can obviously limit the tolls imposed by ISPs. If extensive local monopolies for high speed network access persist, regulation is likely with or without *Bill-Pay*.

The limited willingness of the sender to pay acts as a final incentive to keep tolls lower: if tolls are too high, the sender can choose not to send traffic. Concerns about the ISP's reputation and public image (which affect long term profit prospects) can strengthen the incentive to not impose "unfair" tolls.

Without competition and regulation, the ISPs can charge tolls that amount to monopoly prices and the flexibility of the payment mechanism we propose makes this somewhat easier. While even such an expensive service could still be very valuable to users, we consider monopoly prices an undesirable outcome. Fortunately since competition between ISPs is a reality in many parts of the world where the Internet reaches and monopolies are often regulated, we believe that the basic conditions for the success of *Bill-Pay* are met.

2.2 End-user influenced paths

The incentive structure for *Bill-Pay* works best if users can ensure that their traffic avoids congested and expensive ISPs when alternate paths are available. In addition, a user may want to express other types of preferences: choice between a high latency and a high cost path within an ISP, choice between low jitter and low loss rates, etc. While conveying a limited amount of such information is possible in today's Internet, using the Type of Service field in IP packets, intermediate ISPs have no real incentive to adhere to such user indication. With our incentive-based architecture, user-influenced path and service type selection becomes viable and it can be used to achieve the desired service quality.

We describe here a mechanism that can handle the information exchange between the sender and ISPs. This mechanism uses *opaque alternative descriptors* (OADs) to represent different types of service that users desire and ISPs are able to offer. In particular, there are two kinds of OADs — ISP-OADs with information originating from the ISP and user-OADs with information originating from the sender. ISPs willing to offer choices to the users of *Bill-Pay* packets will use opaque numbers, e.g., choice 1, 2, 3 or 4, to denote the different alternatives. These choices have locally-defined semantics. ISPs mark these available choices in the corresponding bitmap from ISP-OAD fields of the packet's *Bill-Pay* header (see Figure 2). The choice field identifies which specific choice was made for this particular packet. The receiver echoes back an appropriate summary of ISP-OADs together with relevant performance measures to the source of the packet along the reverse path. The sender then uses user-OADs to convey its own preferences to the ISPs along the path. For example, if the sender considers its current path is too expensive for the desired quality, it can communicate this using the undesirable alternatives bitmap of the OAD and mark another alternative in the choice field. Note that the user uses a separate user-OAD for each ISP on the path.

We illustrate the expected operation of OADs using

the example from Figure 1. The first packet from *A* to *B* would have no user-OADs indicated by *A* and it would acquire two ISP-OADs, from ISPs *X* and *Y* on its way to *B*. Let us assume that ISP *X* encodes its choice for this packet as alternative number 1. The acknowledgment from *B* to *A* can return these two ISP-OADs to *A* in the packet payload. If *A* considers the total toll of 14 nanodollars too high, based on its existing knowledge of topology it can ask ISP *X* to forward future packets through ISP *Z* by specifying choice number 2 in user-OADs in subsequent packets. Even if *A* has no prior knowledge of the topology, it can specify that choice number 1 is undesirable and learn about the alternate path with the next packet.

Note that in user-OADs, the sender expresses a preference which is not binding and any ISP is free to ignore the sender's preference altogether. But ISPs have incentives not to do so unless they have a reason to believe that the sender's choice is based on out of date or erroneous information. Also, it is quite possible that a sender does not indicate user-OADs for every ISP on its path. For example, this is the case when there are no alternatives at a given ISP, or when the sender agrees with the default choices of the ISP.

We propose concentrating the end-host's knowledge of network topology and the service quality associated with various choices exposed by ISPs into a module we call the *digital cartographer*. This digital cartographer will have a leading role in picking user-OADs to influence paths and nanopayment levels for future packets in a way that minimizes cost, but achieves the desired service quality. The cartographer's initial knowledge of the network's topology and of the meaning of various choices ISPs expose can come from descriptions published by the ISPs. To build confidence in such information and to keep up to date with changing network conditions, the cartographer would constantly monitor the acknowledgments for *Bill-Pay* traffic to measure actual service quality and toll levels. For individual home users, the digital cartographer is a service running on the end-host, but for larger organizations it makes sense to consolidate this functionality into a campus-wide service that achieves a more detailed understanding of the network by combining information from the transfers of a many end users.

The overhead imposed by OADs is small. Since not all *Bill-Pay* packets need to use them, the header size in most packets can be as small as 4 bytes. Typical AS path lengths in the Internet are 3 and 4, and most are shorter than 7, so a *Bill-Pay* header recording a typical path fits within 20 bytes. Routers can process the packets by inspecting a few fields and writing at most two (the nanopayment amount and the appropriate ISP-OAD) and they need not change the packet size. We believe that this processing can be implemented in the fast path of routers.

2.3 Accounting and authorization

Accounting of nanopayments between different organizations is easy because *Bill-Pay* agreements are local. The basic mechanism required for accounting is a pair of counters for each link connecting two organizations to track the total volume of nanopayments in the two directions. The two organizations can keep separate copies of the counters. At the end of each month or whenever the difference between the two counters reaches a certain value (say $100) the two organizations settle their accounts with actual payments.If the debtor fails to pay, the organization owed money can limit its losses without recourse to law enforcement, by providing no preferential service to future packets from the debtor.

Authorization of nanopayments should be concentrated in a trusted module on the end-host we call the *digital secretary*. Its primary task is to determine how large a nanopayment the user is willing to spend on any given transfer. A large set of initial rules about the importance a typical user assigns to various types of applications helps the digital secretary make decisions, but to build a better understanding of user preferences it requires some initial guidance from the user. We expect that over time, as the secretary learns from the user's answers, it can become sufficiently unobtrusive. The secretary can make small errors by occasionally making small "unjustified" nanopayments to avoid bothering the user with questions. The fact that the secretary does not need an exact understanding of the user's preferences and priorities makes its task more tractable. The digital secretary can also play a role in assembling a "billing statement" that summarizes for the user what he spent his money on. In an enterprise setting the end-host digital secretary would also interact with a central secretary responsible for setting enterprise-wide policies and producing enterprise-wide spending reports.

2.4 Security considerations

If malicious hackers hijack a device that can generate *Bill-Pay* packets to the outside world, they can direct nanopayments to computers they control and cause significant financial damage to the organization the device belongs to. Defenses recognizing suspicious (sudden, large, unusual) nanopayment streams and filtering them out can limit the amount of damage, but we want to disallow such fraudulent payments entirely. Hence, security mechanisms will be an integral part of our proposed architecture. While security mechanisms are needed for all network elements, such as network access devices and routers, in this section we focus on those most vulnerable — the end-hosts.

End-hosts are regularly hijacked by malicious hackers, and we expect them to remain vulnerable for the foreseeable future. Servers that never originate nanopayments,

and those that mirror nanopayments to clients are relatively easy to protect by moving all nanopayment handling into trusted network devices. But clients must be able to originate nanopayments to signal to the network that certain packets are important to the user. The most obvious requirement is to secure the digital secretary and digital cartographer: the attacker should not be able to modify their code or local data, and the digital secretary should be able to interact with the user securely. We can achieve this goal by running the vulnerable operating system and applications inside a virtual machine and placing the secretary and cartographer outside it, or by moving them to a specialized secure device that interacts with the user directly. These trusted modules will need well-specified simple interfaces to interact with applications and protocols running on the end-host. There are two attack models that will gain significance in the proposed architecture.

Impersonation attacks pose a threat to the end-host because a hacker can hijack an application and use it to "impersonate" user behavior and mislead the digital secretary into authorizing unjustified payments. Such threats can be mitigated better if the digital secretary is able to discern regular user behavior from malicious ones. Appropriate research in learning techniques is therefore an important area of future work.

Man-in-the-middle attacks pose a threat because a hacker located on the path between the trusted digital secretary and the ISP can arbitrarily generate new packets or modify packets, including the destination address of packets, and their nanopayments. Such a situation can happen for example if the digital secretary runs on a USB device and the hacker controls the operating system of the end-host. We envision a solution to this problem that involves low-overhead cryptographic checksums, issued by the user's digital secretary and verified by a trusted router at the edge of the network. Similarly, packets that carry sensitive network topology and performance information in the other direction are signed by the router and verified by the secretary. While this would incur additional processing in the data path, we believe that current hardware technologies allow the implementation of such mechanisms in the fast path of enterprise, access, and edge routers.

3 Solutions based on *Bill-Pay*

Once it is adopted by enough ISPs, a network payment architecture such as *Bill-Pay* can contribute to solving many important problems. We briefly sketch one such solution below. The technical report version of this paper [1] also discusses the use of *Bill-Pay* as part of DDoS defenses and the possibility to build micro-payment protocols on top of *Bill-Pay* that can be used (among many other things) to discourage spam.

3.1 Better End-to-end Service Quality

As a first application of *Bill-Pay*, we discuss how an endpoint can achieve the desired service quality in two scenarios: improved throughput for a large transfer (e.g. an unattended download), and low loss rates and delays for time-sensitive traffic (e.g. gaming traffic). The proposed solutions have two main differences with respect to the prevailing view of QoS guarantees for traffic: the price of the transfer is variable, and we rely on active probing instead of explicit negotiation. (Of course, *Bill-Pay* does not provide tight bounds on performance.) The fact that the price of the transfer depends on current network conditions is not a problem if it falls within the amount the user is willing to pay. The extra traffic generated by *Bill-Pay* end-hosts performing active probing is not a problem to the network as they make nanopayments for the probing traffic also.

For large transfers, the overall amount of the nanopayments is likely to be a primary concern and the loss of individual packets, or even large bursts of losses can be acceptable. A reasonable strategy is to not include nanopayments in packets by default. If the performance of the transfer dips below the desired throughput, the sender can choose to add nanopayments. The sender can gradually increase the nanopayments in subsequent packets (while also trying alternate paths) until either the performance of the transfer improves above a threshold or the digital secretary indicates that the limit of the user's willingness to pay has been reached. If the tolls stay lower than the size of the nanopayment for a sustained period of time, the sender decreases the size of the nanopayments.

For time-sensitive traffic, the above strategy is not suitable because a number of important packets can be lost when sudden congestion occurs while the sender is exploring alternative paths and the size of the nanopayment to use. A sender with time-sensitive traffic can discover in advance the paths that provide an acceptable loss rate and delay through active probing with *Bill-Pay* packets done in close cooperation with the local digital cartographer. When the important time-sensitive packets are sent, the amount of the nanopayment is set large enough to get them past typical congestion events (packets may still get dropped, but the probability is much lower). The sender can even exploit the existence of multiple independent paths to increase the probability of timely delivery by sending duplicate copies of important packets along different paths.

4 Related Work

Existing contracts between ISPs either involve flat fees or employ usage-based pricing. Most contracts in the latter category use the 95th percentile traffic volume computed over all 5-minute intervals in a month to determine

how much to charge. Customers pay additional amounts for QoS guarantees. Typically, these contracts are negotiated for several months at a time and the customer can re-negotiate or switch ISPs at the end of the contract period. *Bill-Pay* can be easily implemented by extending existing contracts with a clause that obliges both parties to honor the nanopayments included in the packets they exchange.

Congestion-based pricing for the Internet has been considered in simplified settings [4, 6, 7]. In MacKie-Mason and Varian's "smart market" proposal [4], users include "bids" within packets which indicate their maximum willingness to pay the ISP for access. Gibbens et. al show how smart markets can be realized in practice using simple packet marking mechanisms [2]. In Odlyzko's Paris Metro Pricing [6], an ISP network is divided into several service classes each offering best effort service but at different prices. Traffic classes with higher prices attract lesser traffic, and thus offer improved service.

In general, the above mechanisms work as long as users are vying for access from a single network provider. In contrast, *Bill-Pay* generalizes both the smart markets approach as well as Paris Metro Pricing by allowing users to place "bids" on packets traversing multiple ISPs. Moreover, *Bill-Pay* provides a payment mechanism that can be used to pay remote ISPs without a direct contract.

Micro-payment solutions such as Micali and Rivest's Peppercoin [5] use cryptographic techniques to aggregate very small payments (on the order of cents) into payments large enough to justify the fees associated with money transfers (say $10). Such schemes can be used by network endpoints to perform transactions without any special assistance from the network. Another popular solution is account-based micro-payments such as PayPal [8]. Compared to *Bill-Pay*, these solutions have the advantage of working without support from the network. However, unlike *Bill-Pay*, neither category of solutions can be used to offer fine-grained quality of service in the Internet. This is because such solutions face tremendous scalability challenges when one wants to make payments on the order of a billionth of a dollar on millisecond timescales.

5 Conclusions and Future Work

In this paper, we provided a brief description of *Bill-Pay*, a per-packet nanopayment mechanism based on local bilateral contracts. We believe that *Bill-Pay* is an effective mechanism for providing good end-to-end service between arbitrary endpoints. Furthermore, *Bill-Pay* provides a practical way to solve several other key issues, including DDoS mitigation and spam prevention.

The focus of this paper was to present a broad overview of *Bill-Pay*, its advantages and possible applications. Needless to say, we have left several interesting issues unaddressed. Throughout the paper, we outlined several major open issues; below, we mention a few more:

- What should the optimal behavior of a rational ISP be (e.g., what tolls to set, what service levels to offer)? This might depend on the ISP's topology, traffic patterns and interconnections with peers.
- A related question is how should a rational user "modulate" the size of the payments over the duration of a transfer (and how is this impacted by losses and retransmissions)? And, how do the payments interact with the user's congestion response? E.g. Does the user need to cut the congestion window in half to maintain stability even if he has included a high enough nanopayment in his packets?
- How do the tacit incentives of *Bill-Pay* influence the longer-term growth trends of the network? Will it "automatically" steer the Internet toward richer interconnections between ISPs? Note that *Bill-Pay* implicitly encourages end-users and stub networks to multihome. But will it lead to more path diversity in the rest of the network?
- Does the packet-level routing flexibility of *Bill-Pay* introduce undesirable oscillations into the network? What guidelines should ISPs and end-users adhere to when selecting routes for their traffic so as to ensure stable network operation?

References

[1] C. Estan, A. Akella, and S. Banerjee. Achieving good end-to-end service using Bill-Pay. Technical report, University of Wisconsin-Madison, November 2006. `http://www.cs.wisc.edu/~estan/publications/BillPay.html`.

[2] R. J. Gibbens and F. P. Kelly. Resource pricing and the evolution of congestion control. *Automatica*, 35:1969–1985, 1999.

[3] T. Karagiannis, D. Papagiannaki, and M. Faloutsos. BLINC: Multilevel traffic classification in the dark. In *SIGCOMM*, Aug. 2005.

[4] J. Mackie-Masson and H. Varian. *Public Access to the Internet*, chapter Pricing the Internet. MIT Press, 1995.

[5] S. Micali and R. L. Rivest. Micropayments revisited. In *Cryptography Track at RSA Conference*, 2002.

[6] A. M. Odlyzko. A modest proposal for preventing Internet congestion. Technical report, AT&T Research Lab, 1997.

[7] I. C. Paschalidis and J. Tsitsiklis. Congestion-Dependent Pricing of Network Services. *IEEE/ACN Transactions on Networking*, 2000.

[8] *PayPal.* `http://www.paypal.com`.

[9] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification. In *Internet Measurement Conference*, Oct. 2004.

[10] The utopia consortium. `http://www.utopianet.org/`.

Fighting Coordinated Attackers with Cross-Organizational Information Sharing

Mark Allman[†], *Ethan Blanton*[‡], *Vern Paxson*[†], *Scott Shenker*[†]
[†]*International Computer Science Institute,* [‡]*Purdue University*

ABSTRACT

In this paper we propose an architecture for using cross-organization information sharing to identify members of a group of hosts enslaved for malicious purposes on the Internet. We root our system in so-called "detectives"—savvy network monitors like sophisticated intrusion detection systems or honeyfarms that have a deep understanding of malicious behavior. We augment information from these detectives with observations from a large array of "witnesses" that are already in-place at many locations in the network. These witnesses are not savvy enough to understand that a particular behavior is malicious, but their simple factual observations can be shared with a detective in order to form a broad picture of a group of bad actors. A key aspect of the system is the design of a lightweight mechanism to reliably share enough information between detectives and witnesses to form an understanding of a group of bad actors without sharing more information than necessary, in order to address privacy and competitive concerns.

1 INTRODUCTION

One of the largest current threats to hosts and networks is armies of enslaved hosts ("bots") controlled by a single person or small group. These "botnets" provide an attacker the ability to bring much distributed firepower to bear on a particular target and/or to remain elusive by shifting attacks around the network. The exact procedures for an army of hosts to exchange information and attack other hosts comprise nearly an endless list. Therefore, monitoring the activity of such a group of hosts presents an immense challenge along a number of axes. First, observations from any one point in the network provide only a small view into the overall activity. Second, the vast array of attack vectors and benign communications channels that can be co-opted for control traffic make ferreting out botnet activity very difficult.

To better unmask a group of coordinated attackers we propose a system loosely modeled upon real-world crime fighting. While society employs highly trained crime-fighters ("detectives"), there are not enough such skilled people to monitor all situations where a crime may be committed. As a practical matter, real-world detectives rely on amateurs ("witnesses") who have observations and evidence that aid the detectives in their work. While witnesses are clearly not as skilled and trustworthy as detectives in terms of fighting crime, their value is in their *numbers and prevalence*.

Detectives are charged with detecting patterns of criminal activity, identifying suspects, and then questioning witnesses to fill in the gaps in the detectives' understanding. In particular, we expect detectives to gather information relating to a particular crime—not arbitrary information about arbitrary people or events. Of course, some unrelated information may always "leak" into the process, but anything not germane should be disregarded. Similarly, witnesses should be questioned in such a way that they do not know precisely what they are being asked about, so that they do not learn what criminal activity the detectives are pursuing nor whom the detectives suspect; they only attest to what they have directly observed.

In the realm of fighting groups of coordinated attackers, our detectives are savvy network monitors such as sophisticated intrusion detection systems (IDSs) or honeyfarms [8]. These components of our system can detect "crimes" and discern suspicious patterns of activity. However, as in real life, their viewpoint is too narrow to understand the breadth of activity in disparate corners of the network. Therefore, we also employ general traffic monitors (packet taps, NetFlow logs, proxy cache logs, etc.) as "amateur witnesses" that have evidence to offer, but are themselves not savvy enough to understand that a "crime" has been committed or to put together the complete picture.

In this paper we propose leveraging the *deep understanding* of network detectives and the *broad understanding* of a large number of network witnesses to form a richer understanding of large-scale coordinated attackers. To accomplish this task, we need a way to share information across organizations. Therefore, we offer an information sharing mechanism that (*i*) reveals little-to-no information to anyone who has not witnessed a given event, while still allowing witnesses to provide corroborating evidence and (*ii*) offers the detective reasonable validation that the information from witnesses is sound.

We separate the activity of coordinated attackers into two categories, *attack traffic* and *control traffic*. Attack traffic can range from distributed denial-of-service attacks to scanning for additional vulnerable hosts to re-

cruit into the group. Much research and many products concentrate on finding individual hosts that are actively attacking peers in the network. Control traffic's purpose is twofold: (*i*) for commands to flow from some controller to all members of the group, or (*ii*) for the members of the group to download new malcode or otherwise further prepare for some task (such as an attack). This traffic is more difficult to track than attack traffic precisely because it can appear normal and benign (e.g., simply downloading some data from a URL using HTTP). This normality makes it much harder to identify the traffic as laying the groundwork for an attack. In this paper we focus on using this control traffic to unmask the members in a group of coordinated attackers, even in the absence of an attack.

A high-level example would be a honeyfarm becoming "infected" by a given attack vector and then observing a remote server from which the bots are instructed to retrieve some piece of malcode. The honeyfarm (the detective, in this case) would query witnesses throughout the network for additional hosts that show similar communication patterns. Our information-sharing technique allows the honeyfarm to uncover other hosts that are likely members of the group of coordinated attackers based on witness "testimony", even though these group members and witnesses are scattered throughout the network (such that the honeyfarm cannot directly observe the behavior). Furthermore, unless a witness has observed the activity in question, the honeyfarm's queries about the pattern are obscured such that the honeyfarm reveals little information to the potential witness.

This paper is organized as follows. In § 2 we briefly describe our proposed architecture. Next, § 3 outlines the underlying information-sharing mechanism that enables the system. § 4 outlines related work. We provide brief conclusions and areas for future attention in § 5.

2 ARCHITECTURE

The overall architecture of our proposed system consists of three classes of participants: (*i*) a set D of network detectives (e.g., honeyfarms, sophisticated IDSs, etc.), (*ii*) a set W of witnesses, (*iii*) an aggregation entity that can play the part of a trusted organization like Interpol and gather information from a number of detectives' jurisdictions and then distribute the information to information consumers. The general operation is that some D_i finds a pattern and then interrogates witnesses in search of additional hosts that exhibited the given pattern. From the witness testimony, D_i then forms a list of victims V_i. D_i then sends V_i to the Interpol-like aggregator along with the appropriate pattern. The collector can then gather various V_i sets from various detectives together to form a picture of the group of coordinated attackers. We consider each component of the architecture in turn in the following subsections.

2.1 Detectives

The set of detectives is charged with identifying traffic patterns that correspond to malicious behavior and then querying witnesses to uncover additional hosts that have exhibited the same pattern. The detectives aggregate witness responses and reports the results to the collector.

In § 3.4 we discuss "rogue detectives" who attempt to abuse the system by fabricating patterns in order to "fish" for private information not related to malicious activity. To reduce the risk of such fishing attacks in our system, we keep set D *closed*, i.e., membership is known *a priori* and each host in the set can be readily identified (e.g., using a cryptographic key). Since in our architecture the detectives need to be known and trusted, the set is intended to be kept small (e.g., hundreds of monitors). However, we note that the wealth of information in our system comes from witnesses, not detectives, and therefore a small set of the latter should not present a problem.

An immediate question that a detective must tackle after identifying a suspicious pattern involves determining which witnesses to interrogate. Depending on the situation, the appropriate scope of the queries might range from asking one particular witness a quite-localized question to asking the entire set of witnesses a broad one. For instance, if some group of coordinated attackers employs a centralized code-distribution server then ideally a detective could query a single witness close to the server and reap a wealth of information about which hosts have been seen downloading the code. This might be slightly broadened to a small group of witnesses to account for any of multihomed sites, possible artifacts in the witnesses' logging functions due to their use of sampling, and/or witness misbehavior. The downside of targeted querying is that the detective must assess the role of the witness's proximity to the point of interest. On the other end of the spectrum, if the pattern is not host-specific but along the lines of "incoming connection to port X, outgoing connections to ports Y and Z" then querying as many witnesses as possible around the network will give a more complete picture than trying to query any one particular witness. This is easier to accomplish than targeting witnesses because no notion of proximity is required. In between, there are many possibilities for the querying of various fixed or random sets of witnesses. An in-depth exploration of which witnesses to query is beyond the scope of this paper, but a clear candidate for continued investigation.

2.2 Witnesses

We expect W, the set of witnesses, to consist of a large number (thousands) of simple, general traffic monitoring devices—not particularly designed for security

monitoring—scattered throughout the Internet. Unlike the closed set of detectives, *W* is *open*: new monitors can readily join and witnesses do not need to be vetted before they start answering queries. Since many ISPs and organizations do some sort of general traffic monitoring as a matter of course (for provisioning, debugging, etc.) we aim to leverage these resources rather than rely on additional deployment. That said, these monitors will need to be augmented to answer queries from the detectives in our system. Witnesses are expected to simply log "the facts"—that is, direct observations from the network without any analysis. We do not expect witnesses to "judge" traffic. Rather, the function of witnesses is to provide the detectives with observations to allow a picture of large-scale groups of coordinated attackers to be formed by the detectives. We also note that witnesses in our system can only provide information in response to queries from the detectives and therefore cannot contribute arbitrary data to the system. Witnesses are, of course, also free to ignore requests based on local policy. An incentive for witnesses to contribute information is that the aggregated information will then be made available via the collection and distribution system such that the organization providing the witness will ultimately gain an amplified view of coordinated attackers.

2.3 Collection and Distribution

The Interpol-like collector is a known and trusted entity that aggregates the information collected by the hosts in *D* and makes the information publicly available. Since the members of *D* are well-known, it is tractable to only accept input from trustworthy parties.

The network Interpol serves several key functions. First, it can aggregate information from many detectives to form a more comprehensive picture of groups of coordinated attackers. Second, the collector is responsible for making the results public, but must do so in a way such that the source of each individual piece of information is masked.[1] In addition to collecting and aggregating the information, the collector makes the data publicly available (perhaps via some intermediary distribution points). Doing so allows services to be built that offer the information in myriad ways that operators may find useful. Example services include: simple mirrors of the data via FTP or HTTP, a database server that accepts rich queries, behavioral database entries (ala [1]), or insertion of the data into a robust distributed data structure such as a DHT for reliable dissemination.

Finally, we stress that the collector's role is to aggregate and serve the information, not design the policy. The collector can provide information that will inform policy decisions, but those decisions are left in local hands.

[1] Additional ways to thwart tracking may also be useful to employ, such as Mobile Honeypots [3].

3 Information Sharing Primitive

While the last section sketches our overall architecture, this section focuses on a "loose private matching" scheme to facilitate information exchange between detectives and witness that conforms to the principles outlined in § 1.

3.1 Loose Private Matching

The key idea behind the "loose private matching" mechanism is to enable detectives to encode a query (traffic pattern to look for) in such a way that *(i)* anyone who has actually observed the traffic described by the pattern will be able to recognize it, but *(ii)* the encoding is also ambiguous enough that it could describe a variety of traffic patterns, and therefore it reveals little information to entities that have *not* observed the given traffic pattern. We enforce this distinction by requiring that witnesses who wish to attest to having seen traffic fitting a given pattern must encrypt their responses using the decoded pattern itself as a shared secret. The detective therefore gains a reasonable (not perfect—see below) confidence that the witness indeed observed the traffic in question.

To develop this approach, we consider that patterns being queried are defined by some set of observed actions that a detective can piece together. To illustrate, suppose a honeyfarm *H* is attacked by some host *A* that is scanning for vulnerable hosts to recruit into a group of bad actors via an SQL exploit. Furthermore, after *H* is "infected" it is then asked to TFTP some malcode from code server *C*. A natural pattern a detective might develop from this interaction is "incoming SQL hit from *A* arrives at some host *X*, which in turn initiates an outgoing TFTP request to host *C*". Any *X* (such as *H*) that satisfies this pattern could be assumed to be infected in the same manner as *H*. The pattern could be loosened up such that any communication from host *A* (a known bad actor) could be used instead of just SQL connections to handle attackers that use multiple attack vectors could be found. Or, any TFTP to host *C* could be taken as an indication that the host initiating the connection has been infected. Clearly, these are not iron-clad signatures for an attack, and care must be taken to narrow the scope of queries. For instance, if the malcode happens to have been left on a popular blogging site *B* and infected machines fetch it via HTTP then using the pattern of "HTTP transactions to *B*" is not going to be a useful pattern in finding infected machines.

After forming a pattern, the components of the pattern, $C_1 \ldots C_n$, are then hashed together to form a key, $K = H(C_1, \ldots, C_n)$, which is then used to query witnesses for hosts with similar traffic patterns. The witnesses that receive the query then consult their logs for hosts having communications that match the requested key. Only if the witness has seen the given pattern will it be able

to untangle the given key and provide a useful response. The response is encrypted using the decoded components of K as the shared secret.

Consider an example where a key is constructed by a honeyfarm with a destination IP address d_h, a transport protocol t_h and a destination port number p_h as $K_h = H(d_h, t_h, p_h)$. Now, consider K_h being sent to some number of witnesses with the intent of obtaining a list of source IP addresses that have communicated with hosts in the fashion described in the pattern K_h.

Assume that some witness finds three records that match K_h—with two of these records matching the query sent by the honeyfarm, and the third being a coincidental hash collision. The witness cannot determine which of these matching records (if any), are correct, so all matches are returned. Assume that the source IP address of each matching record s_i is associated with a three-tuple $T_i = \{d_i, t_i, p_i\}$, which represents the material hashed to produce the key matching s_i. In our example, $T_1 = T_2 = \{d_1, t_1, p_1\} = \{d_2, t_2, p_2\}$, and $T_3 = \{d_3, t_3, p_3\}$, a different tuple than T_1 and T_2. The witness forms two responses to be returned to the honeyfarm. The responses consist of a list of addresses d_i from the query followed by each appropriate s_i. These records are encrypted *using T_i as the shared secret*. Specifically, the witness forms the two responses $R_1 = E_{T_1}(\{d_1, s_1, s_2\})$ and $R_2 = E_{T_3}(\{d_3, s_3\})$.

The honeyfarm can now decrypt both responses using $T_h = \{d_h, t_h, p_h\}$ as the shared secret. When decrypting R_1, the honeyfarm will find $d_h = d_1$ as the first address in the list, and so will know that the rest of the addresses in this response are valid for the given query. When similarly decrypting R_2, the honeyfarm will not find d_h as the first item in the returned list, and therefore will know that this response is meaningless and was caused by a collision at the witness. Note that the honeyfarm still does not know either the values d_3 or s_3, as the decryption of R_2 using the inappropriate key $T_h \neq T_3$ yields random data.

We also note that patterns can consist of more than one key that can then be logically connected to form a more specific query. E.g., K_1 may be "source IP A, destination port S" and K_2 may be "destination IP C, destination port T". The query could then be for any host X that the witness observes that satisfies both K_1 and K_2.

3.2 Example

As a concrete example of the above notions, consider a query which requests the source IP address for all hosts having communications that match a pattern that encompasses the destination IP address (4 bytes), transport protocol number (1 byte) and destination port number (2 bytes). From these 7 bytes a key K is formed by taking the product of the bytes (with any zeros rounded up to one). This hash has two crucial properties: (*i*) the hash space is large ($1 \ldots 255^7$, excluding numbers with a prime factor larger than 255) and (*ii*) collisions are guaranteed to be possible in theory. Assuming the protocol number and port number remain the same, IP addresses $a.b.c.d$, $a.c.b.d$ and $a.b/2.c \cdot 2.d$ will all yield the same value (assuming that b and c are even). This ambiguity is critical because it largely prevents anyone who has not seen the corresponding traffic from understanding the question and forming a valid response.

To assess this simple hash function we analyze one day's worth of connection logs from ICSI's border. We used the log from July 27 2006, which consists of roughly 6.2 million connections. We compute a hash using the product of the bytes in the three fields described above for each connection. We find that 11% of the connections hash to a unique K that is not shared by another three-tuple in the dataset. Therefore, 89% of the connections hash to a K with a collision. This indicates that collisions are not just theoretically possible, but ambiguities do in fact naturally occur when using byte-wise multiplication as a simple hash.

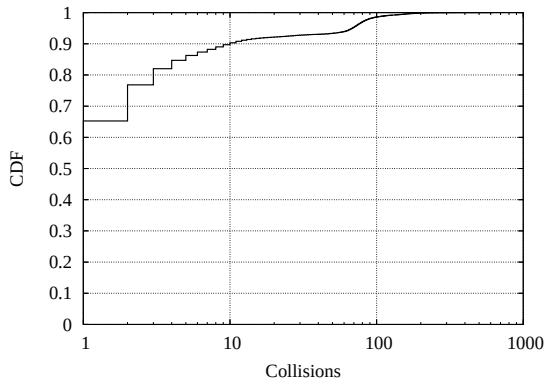


Figure 1: Collisions per key.

Figure 1 shows the distribution of collisions per key in our dataset. The figure shows that roughly two-thirds of the keys are used to cover the 11% of connection that hash to a non-shared K. Further, 90% of the keys correspond to 10 or fewer three-tuples and nearly all keys correspond to 100 or fewer three-tuples. This shows that while there is ambiguity in this particular hash function, the ambiguity likely does not present a logistical problem in transmitting massive amount of data that then needs decrypted by a detective using this hash function. The amount of ambiguity can also be increased with the application of the modulus operator to K such that the size of the hash space is decreased. Alternatively, the ambiguity can potentially decreased by using a smaller window of time such that less traffic is observed.

We stress that we are not proposing this hash as *ideal*. We offer this hash as a simple proof-of-concept that the general idea has promise. For instance, a scheme such as

Private Stream Searching [2] provides many of the desirable qualities we sketch above (and more) at additional computational cost. Crucial future work will clearly involve a survey of alternate hash algorithms and a deeper analysis of the properties of such algorithms.

3.3 Query Language

As described above, the detectives and witnesses have a shared understanding of the components of the queries (IP addresses, port numbers, etc.). While we do not have space to dig into the details of the query language in this paper, we note several possible approaches. First, a standard set of common queries could be defined and a query identifier could be used to synchronize the detectives and witnesses. These fixed hashes could be calculated as the records are initially captured and stored with the records such that a simple lookup on a given key would be straightforward. The downside of such an approach is that the system is locked into a stock set of queries. Another approach is to make the queries self-describing. For instance, the queries may come with a bit mask indicating which components from the traffic are included in the hash. This is more flexible than a system with a standard set of queries, at the price of computing a hash for every stored record every time a query arrives. A third approach is a hybrid—with a set of common questions and a self-describing mechanism for richer queries.

3.4 Cheating

3.4.1 Detectives

The fundamental way that detectives can cheat is to fabricate a query to fish for private information. For instance, a query could easily be constructed that asks for hosts that accessed some unsavory web server. This is essentially inherent in the mechanism. Even if the queries include additional hard-to-fabricate evidence that network traffic has been observed (e.g., a TCP initial sequence number as "proof of standing") and can be verified by a witness,[2] the bar for cheating is only slightly raised. The detective then only needs to observe or execute some access to the resource in question to then gain the appropriate credentials to fish for a broader set of private information. One way to possibly mitigate the impact of fishing attacks is for witnesses to not answer queries about some pattern until a number of independent detectives have requested information about the same pattern. This offers some assurance that a rogue detective is not simply trying to coax witnesses to send private information that has no relevance to attacks. The nature of our architecture aids the mitigation of this fishing attack, as well, because we intend the system to consist of a fairly small number of detectives and to gain the bulk of the information about the members of groups of coordinated attackers from witnesses. Therefore, as sketched in § 2 the set of detectives is known and assumed trustworthy in our architecture (with the caveat that witnesses can clearly further constrain detectives using the thresholding approach sketched above).

[2] This would limit the witnesses that can be queried to those along the path of the specific observed traffic that the detective describes.

3.4.2 Witnesses

Witnesses can either withhold information or fabricate information in response to queries. Our envisioned system includes a large number of witnesses with many vantage points that are likely overlapping. Therefore, the fact that some witness W_i withholds some record does not mean that another witness W_j will not furnish that record to the querying detective. A witness can also try to inject bogus records into a response in two ways. First, bogus records could be piggybacked on a legitimate response. In other words, the witness was able to untangle the query by looking through the local logs, but then instead of simply reporting legitimate log contents, either a completely bogus list or a partially bogus list is returned in an attempt to implicate innocent actors. This can be effectively mitigated within our proposed system by collecting multiple independent witness statements about some actor before making a decision. Another variant of the injection attack is to attempt to crack the hash and respond to a query despite having not seen corresponding traffic. In this case, the chances of the "witness" guessing the correct components the detective used to form a hash are dependent on the hash function, and with an appropriate hash function this should be quite difficult. Further, enough ambiguity in the hash should be in place such that a brute force responding with all possible combinations of the initial components should be readily apparent.

A final form of attack, difficult to defend against, is when an adversary is able to correlate across multiple queries (either made to multiple witnesses, or a succession of queries made to the same witness) to infer what information a detective seeks. Even without multiple queries, an adversary can make some inferences in this regard by inspecting the witness records that match a given query and assessing which matches likely reflect more interesting behavior than others.

4 Related Work

Sharing information across networks and organizations to aid security is not a novel concept. [7] outlines a system that allows sharing of information across local or remote instances of the Bro IDS. The system relies on prearranged certificates to authenticate peers and can scope the information being shared with each peer. The scheme presented in this paper does not require pre-shared certificates, nor do all components of the system need to be

well-known, but the information shared is also not as rich as two IDSs could agree to share.

A general "private matching" approach is given in [4] whereby two encryption functions exist such that $E_1(E_2(x)) = E_2(E_1(x))$. Two peers can exchange their encrypted version of some x and determine if x is the same without revealing x. We use a loose form of private matching that does not rely on a shared understanding of encryption functions or keys, which hinders scalability.

The SPIE system [6] allows victims of network attacks to trace the attack back to its origin without relying on the (possibly spoofed) IP address by having routers keep a history of all the packets forwarded (in a Bloom filter). This history can then be queried by producing one of the attack packets—with routers indicating whether or not they have forwarded the given packet. This is an instance of the system we propose in this paper. However, we expand the notion to include less specific questions and non-binary answers.

[1] proposes a system that allows for the reporting of malicious hosts into an open database for anyone to use in forming policy decisions (e.g., "host X is a scanner"). The validity of the information in the database is left to the consumer's assessment of the information provider's reputation. The system assumes a large number of intelligent monitors to collect wide-scale information, whereas we focus on leveraging information from generic network monitors that cannot make behavioral judgments on their own.

Finally, [5] suggests that, rather than looking at the low-level details of communications (e.g., payloads of IRC packets), large groups of coordinated attackers can be identified by deriving communication patterns in the form of a who-talks-to-who "contact graph" from an aggregate view of the traffic (or, even a subset of the aggregate view). Our proposed scheme is similar, but aims to form notions of botnets by using observations of malicious activity to chase down similar activity elsewhere. Further, we offer a scheme that allows pertinent information exchange and does not rely on arbitrary traffic data.

5 Conclusions and Future Work

In this paper, we make several contributions. First, we offer a system that leverages existing wide-scale generic traffic monitoring ("witnesses") to aid a much smaller group of intelligent systems ("detectives") in forming both a *deep* and *broad* understanding of groups of coordinated attackers that can then be used network-wide. Second, we offer a framework for identifying coordinated attackers that does not rely on the specifics of the way any particular botnet operates and therefore may provide a longer shelf-life than schemes that rely on intimate knowledge of botnet behavior. Finally, we sketch a loose private matching mechanism to allow for information sharing about mutually observed network events. The mechanism has promise within the system we have outlined, in addition to possible other tasks where scoped sharing of data across organizations is useful.

While we believe this paper offers a number of novel contributions, much additional work on nearly all aspects of the proposed system is required. For instance, further investigation into hash functions for use within the loose private matching scheme is needed. In addition, an investigation into deriving activity patterns is required such that honeyfarms can compute these patterns on the fly. Such an investigation will also provide information about the critical components of the patterns, which will aid in the design of a query language that detectives and witnesses can share. Finally, a large number of logistical questions remain, such as, will operators find sharing in the fashion presented in this paper is reasonably safe given the potential benefits?

Acknowledgments

Discussions with a large group of people have helped our thinking along, including Marina Blanton, Jason Franklin, David Lapsley, Carl Livadas, Doug Maughn, Robin Sommer, Tim Strayer and Robert Walsh. This work was supported in part by the National Science Foundation under grants ITR/ANI-0205519, NSF-0433702 and STI-0334088, for which we are grateful.

References

[1] M. Allman, E. Blanton, and V. Paxson. An Architecture for Developing Behavioral History. In *Proceedings of USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet*, July 2005.

[2] J. Bethencourt, D. Song, and B. Waters. New Constructions and Practical Applications for Private Stream Searching (Extended Abstract). In *IEEE Symposium on Security and Privacy*, May 2006.

[3] B. Krishnamurthy. Mohonk: Mobile honeypots to Trace Unwanted Traffic Early. In *ACM SIGCOMM Network Troubleshooting Workshop*, Sept. 2004.

[4] Y. Li, J. Tygar, and J. Hellerstein. Private Matching. *Computer Security in the 21st Century*, pages 25–50, 2005.

[5] V. Sekar, Y. Xie, D. Maltz, M. Reiter, and H. Zhang. Toward a Framework for Internet Forensic Analysis. In *Proceedings of ACM SIGCOMM HotNets*, 2004.

[6] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer. Single-IP Packet Traceback. *IEEE/ACM Transactions on Networking*, 10(6):721–734, Dec. 2002.

[7] R. Sommer and V. Paxson. Exploiting Independent State For Network Intrusion Detection. In *Proceedings of ACSAC*, 2005.

[8] M. Vrable, J. Ma, J. Chen, D. Moore, E. VandeKieft, A. Snoeren, G. Voelker, and S. Savage. Scalability, Fidelity and Containment in the Potemkin Virtual Honeyfarm. In *ACM Symposium on Operating System Principles*, Oct. 2005.

Black box anomaly detection: is it utopian?

Shobha Venkataraman*, Juan Caballero*, Dawn Song*, Avrim Blum*, Jennifer Yates†
*Carnegie Mellon University †AT&T Labs-Research

ABSTRACT

Automatic identification of anomalies on network data is a problem of fundamental interest to ISPs to diagnose incipient problems in their networks. ISPs gather diverse data sources from the network for monitoring, diagnostics or provisioning tasks. Finding anomalies in this data is a huge challenge due to the volume of the data collected, the number and diversity of data sources and the diversity of anomalies to be detected.

In this paper we introduce a framework for anomaly detection that allows the construction of a black box anomaly detector. This anomaly detector can be used for automatically finding anomalies with minimal human intervention. Our framework also allows us to deal with the different types of data sources collected from the network. We have developed a prototype of this framework, TrafficComber, and we are in the process of evaluating it using the data in the warehouse of a tier-1 ISP.

1 INTRODUCTION

ISPs collect large amounts of data from their networks into warehouses and use this information for provisioning, analysis and generally to guarantee the health of the network. Given this wealth of information, ISPs are interested in using anomaly detection techniques on this collected data to diagnose incipient problems before they can significantly impact the network.

The network data collected varies depending on the ISP's storage resources and monitoring capabilities but is generally characterized by its volume and diversity. The volume of the data collected, which can be in the order of gigabytes per day for a large national ISP, makes manual analysis of the data infeasible. In addition to the volume, the diversity of the data is also daunting. An ISP could expend significant time studying and modeling one feature from a single data source and only gain insight about a drop in a sea of anomalies. ISPs need a scalable approach that automates this process and requires no previous knowledge or analysis of the behavior of the data.

ISPs currently collect measurements such as: byte counts, link error counts and CPU utilization from SNMP data; periodic snapshots of the network topology; system logs from the servers and routers; end to end path measurements such as loss or delay; physical measurements such as current through an optical amplifier; configuration files; and many others. Previous approaches for anomaly detection have usually focused on a small set of data sources, usually packet traces or SNMP data, and a small set of features like volume, byte counts, or IP header features [2, 5, 7, 8, 18, 20, 23, 24]. This kind of approach only explores a few points in the anomaly detection space and would require an ISP to operate multiple specific anomaly detection systems working in parallel, requiring network analysts to work with different outputs from each system, resulting in high operating costs.

In this paper we propose a different approach. We seek answers to the following question: to what extent can we build a black box anomaly detector that automatically finds anomalies in any data source and any feature gathered from the network? This approach would allow an ISP to deal with the volume and diversity of the collected data in a scalable and comprehensive way. We introduce a general framework that splits the problem into two: 1) transforming multiple data sources into a common input and 2) building a black box anomaly detector that, given that input, automatically outputs multiple types of anomalies. In this paper, we tackle the following problems:

Finding novel anomalies not yet seen in the data: The traditional approach to automatic detection of anomalies is to use machine learning algorithms on samples of both normal data and anomalies, and train a good classifier to separate these samples [4, 8, 13, 16]. The basic problem with this approach is that it limits us to the detection of anomalies *which are already present in the samples*. Thus, our approach is different – we aim to find features of the data that consistently behave in the same way in normal data. Then, when we see new, possibly mixed data, we can use this behavior to decide if the data is normal or contains anomalies. In machine learning terminology, we view the problem as *learning only with positive examples* [17].

Identifying features of interest: Given a data source, we are interested in detecting changes in the behavior of the data source. One could use several features from the same data source for detecting the change but not all of them may be equally useful. For example, it is difficult to detect anomalies in a feature that exhibits a lot of fluctuation and very little regularity in normal data. Our framework automates the exploration of multiple features from the same data source and the identification of those that have a consistent behavior in the normal data, according to the models and metrics defined, and thus can be used by the black box anomaly detector to detect changes in the data source.

Detecting anomalies without domain knowledge: Given a set of features of interest, our black box anomaly detector finds anomalies in those features, using no apriori domain knowledge about the behavior of those features. This is done by automatically extracting the ex-

pected behavior from the normal data and then flagging significant deviations from the expected behavior as anomalies. Thus, our anomaly detector flags changes in behavior with respect to the training data.

Handling multiple data source types: Our framework takes into account the diversity of data sources currently collected by ISPs and how they can be transformed into the proper input for the black box anomaly detector.

Gradually increasing the scope of the detection: The black box anomaly detector provides feedback on when a model does not apply to a feature, which allows us to gradually add new models as they are needed. Since each model allows to detect specific families of anomalies, it also allows us to gradually add new models that allow detection of more sophisticated families of anomalies.

2 FRAMEWORK OVERVIEW

We split the question of how to build a black box anomaly detector that handles multiple data sources and features into two smaller questions, that we address in turn. The first one is: can we transform the different types of data sources collected from the network into a common input that can be used by our black box anomaly detector? The second one is: how do we build that black box anomaly detector? In Section 3 we deal with the issue of different data source types and then in Section 4 we explain how to build the black box anomaly detector.

Now, we briefly introduce the four components of our framework: transformations, features, models and metrics. Transformations allow us to convert from data sources of different types to features of a single type. Features represent characteristics of the data that we are interested in. In our framework they are instantiated as a time series of real values that can be used as input to our black box anomaly detector. For example, a data source might be a packet trace captured from the network. From this data source we can extract multiple features such as the traffic volume per connection or the number of destinations contacted by every source on port 80. We discuss both transformations and features in Section 3.

A model is an abstraction that allows us to represent some property of the data. For example, we can model the distribution of a feature. Finally, the metrics are used to evaluate how good the model is representing that property of the data and also allow us to find deviations from the model, that is anomalies. We discuss both models and metrics in Section 4.

3 HANDLING DATA SOURCE TYPES

In this section we address the question of how to transform the different types of data sources collected from the network into a common input type, that can be used by our black box anomaly detector.

3.1 Data source type classification

A time series is a time-ordered sequence of data points. We classify time series into four different classes according to two properties: data point values and data point time spacing. With respect to the data point values, we classify a time series into *real-valued*, when the data point is a point in $\Re^k$, or *structured* when the data point is a more complex structure. With respect to the data point time spacing, we classify a time series into *constant-spaced* when data points are equally spaced, or *variable-spaced* when the spacing between data points is not constant.

	Constant-spaced	Variable-spaced
Real-valued	SNMP measurements End-to-End path measurements	SNMP measurements with missing data
Structured	Periodically sampled topology information	Packet traces Netflow logs Configuration files Syslogs

Figure 1: Data source type classification

We consider that data sources can be associated with timestamps, e.g. of the time when they are collected. Thus, a data source can be considered a time series and classified into one of the four classes shown in Figure 1. We can then define functions that convert between the different classes of time series, which we describe in Section 3.3.

3.2 Features

We define a *feature* to be a representation of some network characteristic that is instantiated as a k-dimensional real-valued time series. Some of the data sources gathered from the network can be used as features themselves but others cannot. For example, a common SNMP measurement such as packets per second on a link, collected every 5 minutes, is a data source and can also be used as a feature, since it represents a network characteristic and has the format of a real-valued constant-spaced time series. On the other hand, a packet trace is a data source but is not a feature. In fact, many features can be extracted from this data source such as the traffic volume per connection or the number of destinations contacted by every source on a specific port.

Note that the input to the black box anomaly detector is a feature represented as a k-dimensional time series. In our current implementation, we focus on modeling features that are represented as a one-dimensional real-valued time series. However, the framework allows the anomaly detector to take as input k-dimensional time series (which we could use to represent graphs, matrices, etc.) in order to support more complex models [3].

3.3 Transformations

We define a *transformation* as a function that takes as input a time series, and outputs another time series. We are interested in transformations that take as input a time series belonging to one of the four classes shown in Figure 1 and output a time series that belongs to a different class. Then, when given a data source we can define a sequence of transformations that will extract a feature represented as a real-valued time series.

This allows us to reduce the problem of anomaly detection with different data source types to finding the proper sequence of transformations, for each data source, and dealing with features of a single data type.

Whether the input time series needs to be constant-spaced or variable-spaced depends on how the anomaly

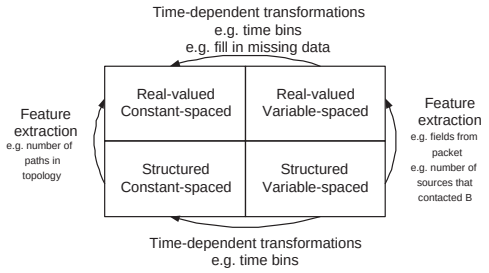


Figure 2: Example data sources transformations

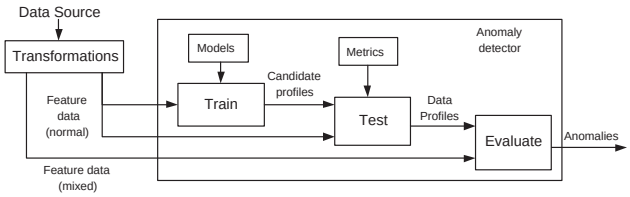


Figure 3: TrafficComber Overview

detector uses it. For example, we might be interested in removing periodic spikes because they hide other smaller spikes of interest, and then a constant-spaced time series will be needed. But if the model simply defines an upper bound on the average then the input time series can be either constant-spaced or variable-spaced.

We can divide transformations into two groups: time-dependent transformations and feature-extraction transformations. Time-dependent transformations operate within a single row of the matrix shown in Figure 1. As examples of time-dependent transformations, we describe two that we currently use to convert time series from variable-spaced to constant-spaced. The first is a bin transformation that divides time into equal size intervals, and bins together data points falling into each interval. The other is a generic missing data transformation, designed to fill in missing data points in the series [10].

Feature-dependent transformations operate within a single column of the matrix shown in Figure 1. They allow the extraction of network characteristics from more complex structures such as netflow logs, configuration files or packet traces. Figure 2 shows the different groups of transformations.

4 BLACK BOX ANOMALY DETECTOR

In Section 3, we have explained how to transform the data source into a set of real-valued time series, that we use as input to the blackbox anomaly detector. We now discuss how to build the blackbox anomaly detector.

4.1 Overview

An anomaly is a deviation from an expected behavior. This naturally poses a fundamental question: do we know the expected behavior of our data? In other words, can we predict the behavior of our data?

One approach to anomaly detection is to compare the given data to some domain knowledge of how the data should behave. However, this approach does not scale well, and is especially unsatisfactory with network data, where most often we do not know how the data should behave or even when we think we do, experimentation often proves us wrong. Without any domain knowledge on the data, we need to extract the expected behavior from the data itself.

Another approach is to analyze the behavior of the data, computing some measurements on the data, and examine how these measurements hold in future data. However, not every measurement will be applicable to future data. How do we know which ones, if any, will be applicable and indicative of future data? For example, we could measure the maximum value that appears in the normal data and use that value as an upper bound on future data. But how do we know whether that specific value will really be an upper bound? In order to be able to relate the past measurements with the future, we need to know what the relationship is between the data seen so far and the future data. Since we do not use domain knowledge about the data, we make the assumption that a specific property holds on the data, and then we test if the assumption is true, that is, if the data really behaves as implied by that underlying property.

Our approach is the following: we *search* for general properties of the data, such as independence across time intervals, and build models of the data based on these properties. For example, one model we could build is an upper bound on the data values, while another model could be an upper bound on the 50th percentile value of the distribution. Then, we assume that the data follows that model and *test* if this assumption is true by verifying if the data agrees with the model according to some metrics. If it does not, then the underlying property used to build the model does not hold and we need to test another model, built on a different property. If the model holds, then we can assume, to the extent tested, that the underlying property used to build the model holds on the data. We call this a *search-and-test* approach because we search for general properties of the data and test if they hold.

4.2 Train, Test, Evaluate

Our approach for implementing the black box anomaly detector is shown in Figure 3. We begin with a class of models and a set of normal data in the form of a time series of real values. For each model that we want to test, we use the input normal data to compute the values of the model parameters. We call the parameterized models *candidate profiles* and refer to this as the *Training* phase.

Next, in the *Testing* phase, we assess the candidate profiles on a new set of normal data. The candidate profiles are tested using metrics related to the model and only candidate profiles that satisfy the metrics are kept, the rest are discarded. Given sufficient normal traffic data, this approach will automatically generate profiles that characterize the input data and automatically discard all profiles which do not.

The data profiles generated through Training and Testing can then be used during the *Evaluation* phase for real-time anomaly detection, by applying them on a set of mixed data and flagging any deviations from the profile that the mixed data might present.

4.3 Modeling the data

In this section we describe the last two components of our framework: the models with their underlying assumptions and the evaluation metrics.

4.3.1 Models and Assumptions

In order to extract the profile of the traffic and have some confidence bounds on its prediction, we need to have some assumptions on the relationship between the data that we have seen so far, and the data that we expect to see in the future. Otherwise, the profile cannot imply anything about the new data. For example, each month's traffic might be drawn *iid* from a distribution over months, that has a 50% chance of being a high-traffic month, and 50% chance of being a low-traffic month. Clearly, in this case, training over a month of traffic would not yield a useful model over the next month.

With each assumption that we consider, we can define a class of models based on that assumption. We then build the models by computing appropriate values for the model parameters, and showing that these computed parameters are *tight*, i.e., they are not too far from the (unknown) true parameter values. For every model built for a given feature, we now need to test that the relevant underlying assumptions hold; if they do not, the model is not valid. We handle this in the next section with the last component of our framework, the evaluation metrics. [1]

There are always tradeoffs associated with the choice of assumptions to use for the data. If we choose models with very strong assumptions, the data may not obey these assumptions. On the other hand, if we restrict ourselves to models with very weak assumptions, the guarantees we would get from the model would be very weak. For example, if the only assumption that we make is that the values have an upper bound, then any guarantee we can make on the values can only involve this upper bound. The rest of the distribution might change significantly, but we would not be able to detect it with models involving only this assumption.

Therefore, we explore models based on a range of assumptions in our anomaly detector, and we build models for the features starting from the strongest assumptions, weakening them as needed to fit the data.

4.3.2 Evaluation Metrics

We need to be able to evaluate the candidate profiles in order to check if they fit the data, that is to test the validity of the model assumptions they are based on. In addition, we need to evaluate how likely a deviation from the model is to determine an anomaly.

For every model, we can define some properties that should be satisfied by feature values that fit this model. We refer to these as *model evaluation properties*. Some example model evaluation properties are the mean, the variance and the 90% percentile. There may not be a small (or even finite) set of properties that are sufficient to ensure that the model holds, so we may have to pick a subset of these properties to examine.

[1] Note that it is not always possible to say when an assumption does hold; the best we can sometimes say is whether data is consistent with the assumption.

In order to evaluate the model, we examine the values of these properties, and test how likely they are to have been generated from the model that we are testing. Thus, our *evaluation metrics* for the model are the likelihoods of the evaluation properties. So, for example, when we have built a model and our evaluation property for testing it is the variance, we compute the empirical variance on the data, and test how likely it is that this value of the variance was generated from the model that we built.

Thus, our evaluation of the validity of the model can be only as good as the properties of the model that we consider. For this reason, we will ask whether a model and the relevant evaluation property *together* are valid for the feature, rather than ask if a particular model is valid.

Since we can have multiple evaluation properties and associated metrics for a particular model, we may find that some of them are violated while others are never violated. If any of the evaluation properties do not hold during the testing phase, this indicates that the underlying assumption generating the model is violated and that model does not represent the feature accurately.

Thus, the space of anomalies we can detect is defined by the classes of models and their evaluation metrics that we use in the anomaly detector. These allow us to gradually increase the space of anomalies the detector can detect, e.g. as more sophisticated models and metrics are added to the anomaly detector, more families of anomalies will be detected. Also, it allows a different evaluation of the models: the larger the family of anomalies that a model can detect, the more suited it is for anomaly detection.

4.3.3 Application of the Framework

As a concrete example of the application of our framework, we now discuss some of the assumptions, models and the metrics that we consider.

We explore models that work on a single feature of one-dimensional real-values and consider two examples of assumptions on the values: *interval-independence* and *source-independence*. For the interval-independence assumption, we assume that each value in the time series is generated *iid* from the model (so, independently of the other values of the time series). For the source-independence assumption, we assume that each value of the time series is generated as a sum of independent processes. For example, the time series may represent the number of sources that are active on a particular port, and it may be reasonable to assume that each source acts independently of the other sources. With these two assumptions, we build two models: the first uses only interval independence, while the second uses both.

When we use only the first assumption, the data values could come from any distribution. So, the model we build consists of upper bounds for the various percentiles of the distribution. That is, we compute an upper bound for say, the 50th percentile, 70th percentile, etc. of the distribution. We evaluate this model by examining if the upper bound of each percentile is obeyed (under probabilistic guarantees). We refer to this model as the *percentile model*.

When we use both assumptions, the class of distributions the values can belong to is the generalized binomial distribution. The model we build is an estimate of the relevant parameters of the generalized binomial distribution. The evaluation properties we examine are properties that hold on this distribution, and we test how likely the property observed on the data is to come from a binomial distribution with the estimated parameters. We refer to this model as the *coin-tossing model*.

While the percentile model is clearly more general than the coin-tossing model, the guarantees we can get from the percentile model are weaker than those we can get from the coin-tossing model. For example, in the percentile model, any single value could be arbitrarily high without being anomalous (e.g., we may have an estimate on the 90th percentile of the distribution, but any individual value could be arbitrarily large), whereas in the coin-tossing model, we estimate how likely a particular value is to come from the distribution.

These two models capture many aspects of the behaviour of a feature, and so metrics can be defined to find changes in these aspects of the behaviour. For example, some data sources exhibit periodicity: there is a spike in their value periodically. As long as the distribution itself does not change, this can be modelled using the percentile model. However, there are other patterns of behaviour that these models cannot capture. For example, if the normal data exhibits an increasing trend, then the distribution of the values is no longer fixed, and therefore, we cannot model it with the coin-tossing or the percentile models. We plan to explore models that are able to capture these trends using non-parametric models, time series analysis and forecasting.

5 Related Work

There has been a wealth of research on anomaly detection. We focus here on the work that we feel is the closest to ours. One line of previous work has focused on specific features. McDaniel et al. [12] proposed profiles that capture the set of peers with which a host communicates and used the profiles for worm containment. Lakhina et al. [7] compute the entropy of the distribution of different IP header fields, and use it for automatically classifying network anomalies through unsupervised learning. LERAD [11] use a different approach that considers each byte of a packet header as a different attribute. Barford et al [2] use wavelet analysis to find anomalies. Thottan et al [20] propose a statistical signal processing technique to detect abrupt changes. Another line of previous work has presented frameworks for anomaly detection. Lee et al. [8] presented a framework that uses both normal and anomalous data to find the characteristic features of anomalies. Zhang et al. [24] presented a framework for network anomography that builds in the linear relationship between link loads and traffic matrices. Our work differs in that our framework considers different data sources types, uses only normal data, and allows for different features and models with different kinds of assumptions to be used for anomaly detection.

There have also been a number of studies that applied network profiles to intrusion detection. EMERALD [15] combines signature analysis with statistical profiling, MINDS [9] and ADAM [1] use data mining techniques to build profiles through learning network connections assumed to be normal. SPADE [19] generates a packet table based on connection history for each port and hosts and raise an alarm for packets rarely seen. eBayes TCP [21] employs Bayesian inference to categorize connections into pre-defined models, and WSARE [22] adds temporal attributes which allow to detect periodic and seasonal anomalies. Finally, Pang et al. [14] and Labovitz et al. [6] show anomalies present in different types of network data.

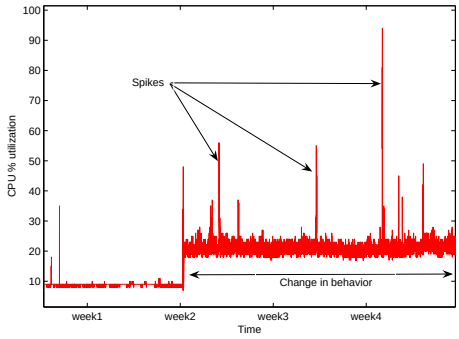


Figure 4: One month of CPU utilization in a backbone router

6 Preliminary Results

We have implemented a prototype of our framework, TrafficComber, and we are currently in the process of evaluating it.

Data sources and features We have started evaluation using two different data sources: packet traces captured at the border router of a departmental network and SNMP measurements from the network of a tier-1 ISP.

As a proof-of-concept, we are currently testing two features extracted from the packet traces: (1) the number of sources contacting more than k destinations on a fixed port, and (2) the number of ports in which a fixed source contacts more than k destinations. We refer to these features as *port* and *src* respectively. The first feature looks at the outbound traffic on a single port aggregated over all of the hosts in the network. The second feature models the outbound traffic of each individual host in the network. Both features aim to detect events characterized by one or a few sources having large fan-out. Some example applications of these features could be detecting worm outbreaks or finding hosts with heavy P2P usage located inside the monitored network.

Given the large number of active hosts and ports in the network, a manual approach is infeasible. For example, the number of hosts in our network is above 1000 and so far the number of active hosts for which we have automatically built profiles varies from 300 to 975 hosts depending on the feature and time period.

From the SNMP measurements, we are currently modeling the router CPU utilization. Figure 4 shows one month of CPU utilization from a backbone router. It in-

Data Source	Feature	% invalid models
Packet traces	port	3%
Packet traces	src	7%
SNMP measurements	cpu	25%

Table 1: Percentage of invalid models when applying the coin-tossing model on different features

cludes two types of anomalies: spikes and changes in behavior. We have found that changes in the behavior of the CPU utilization are widespread among the backbone routers and so far we have identified two main causes: software upgrades and hardware replacements.

Models and metrics To validate that the anomaly detector truly tells us which models are valid for a specific feature, we apply the coin-tossing model introduced in Section 4.3.3 to the three different features. We know that the coin-tossing model uses a source-independence assumption that does not really apply to the cpu and src features. If the system determines that the model does not fit the data it will discard the model.

Table 1 shows how often the anomaly detector discarded models when applying the coin-tossing model for the different features. Note that 25% of models were discarded for the cpu feature, implying that the coin-tossing model is not able to create valid profiles for that fraction of the routers. This in turn suggests that it would be better to use a different model (with different assumptions) for this feature, and we want to evaluate other models for this feature. The fact that the anomaly detector is able to tell us when a model does not apply to a feature allows us to gradually add new models as they are needed.

7 CONCLUSION

Anomaly detection is a fundamental tool for ISPs to maintain the health of their networks. But the volume and diversity of the data currently gathered from the network requires a comprehensive and automatic approach, rather than a set of individual solutions. In this paper we have shown that it is possible to build a black box anomaly detector that handles the diversity of data sources and features collected from the network. We have introduced a framework that splits the problem into two: handling different data sources and building a black box anomaly detector. We deal with diverse data sources through sequences of transformations that convert them into sets of features with a well defined data type. Then, our search-and-test black box anomaly detector automatically tests for the presence of underlying properties in those features that allow us to detect changes in the behavior. This approach allows us to detect novel anomalies not yet seen in the data and to explore the multiple features of interest, while gradually increasing the scope of the detection. We have developed a prototype of our framework, TrafficComber, and we are in the process of evaluating it using the data in the warehouse of a tier one ISP.

8 ACKNOWLEDGEMENTS

We would like to thank Tamraparni Dasu, Zihui Ge, Ajay Mahimkar, Eric Vance and Jia Wang for many discussions; Min Gyung Kang and Pongsin Poosankam for their help with the project, and Vyas Sekar and the anonymous reviewers for their insightful comments.

REFERENCES

[1] D. Barbara, J. Couto, S. Jajodia, L. Popyack, and N. Wu. Adam: detecting intrusions by data mining. *IEEE Workshop on Information Assurance and Security 2001*.

[2] P. Barford, J. Kline, D. Plonka, and A. Ron. A Signal Analysis of Network Traffic Anomalies. *Internet Measurement Workshop 2002*.

[3] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. An Information-Theoretic Approach to Detecting Changes in Multi-Dimensional Data Streams. *38th Symposium on the Interface of Statistics, Computing Science, and Applications*.

[4] G. Giacinto and F. Roli. Intrusion Detection in Computer Networks by Multiple Classifier Systems. *International Conference on Pattern Recognition 2002*.

[5] Y. Gu, A. McCallum, and D. Towsley. Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation. *Internet Measurement Conference 2005*.

[6] C. Labovitz, G.R. Malan, and F. Jahanian. Internet Routing Instability. *ACM SIGCOMM 1997*.

[7] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. *ACM SIGCOMM 2005*.

[8] W. Lee and S. Stolfo. A Framework for Constructing Features and Models for Intrusion Detection. *ACM Transactions on Information and System Security*, **3**(4).

[9] L.Ertz, E. Eilertson, A. Lazarevic, P. Tan, J. Srivastava, V. Kumar, and P. Dokas. The MINDS - Minnesota Intrusion Detection System. In *Next Generation Data Mining*. MIT Press, 2004.

[10] R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, 1987.

[11] M. V. Mahoney and P. K. Chan. Learning Rules for Anomaly Detection of Hostile Network Traffic. *Third IEEE International Conference on Data Mining*.

[12] P. McDaniel, S. Sen, O. Spatscheck, J. Van der Merwe, B. Aiello, and C. Kalmanek. Enterprise Security: A Community of Interest Based Approach. *Network and Distributed Systems Security 2006*.

[13] A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. *ACM SIGMETRICS 2005*.

[14] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson. Characteristics of Internet Background Radiation. *ACM Internet Measurement Conference 2004*.

[15] P. A. Porras and P. G. Neuman. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. *National Information Systems Security Conference 1997*.

[16] M. Sabhnani and G. Serpen. Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context. *International Conference on Machine Learning, Models, Technologies and Applications 2003*.

[17] H. Shvaytser. A Necessary Condition for Learning from Positive Examples. *Machine Learning journal*, **5**(1):101-113.

[18] A. Soule, K. Salamatian, and N. Taft. Combining Filtering and Statistical Methods for Anomaly Detection. *Internet Measurement Conference 2005*.

[19] S. Staniford, J. A. Hoagland, and J. M. McAlerney. Practical automated detection of stealthy portscans. *7th ACM Conference on Computer and Communications Security*.

[20] M. Thottan and C. Ji. Anomaly Detection in IP Networks. *IEEE Transactions on Signal Processing*, **51**(8).

[21] A. Valdes and K. Skinner. Adaptive, Model-Based Monitoring for Cyber Attack Detection. *Third International Workshop on Recent Advances in Intrusion Detection*.

[22] W. Wong, A. Moore, G. Cooper, and M. Wagner. Bayesian Network Anomaly Pattern Detection for Disease Outbreaks. *Twentieth International Conference on Machine Learning*.

[23] K. Xu, Z. Zhang, and S. Bhattacharyya. Profiling internet backbone traffic: behavior models and applications. *ACM SIGCOMM 2005*.

[24] Y. Zhang, Z. Ge, M. Roughan, and A. Greenberg. Network anomography. *Internet Measurement Conference 2005*.

Glavlit: Preventing Exfiltration at Wire Speed

Nabil Schear, Carmelo Kintana, Qing Zhang, Amin Vahdat
Department of Computer Science and Engineering, University of California at San Diego
{nschear, ckintana, qzhang, vahdat}@cs.ucsd.edu

ABSTRACT

Protecting sensitive data is no longer a problem restricted to governments whose national security is at stake. With ubiquitous Internet connectivity, it is challenging to secure a network – not only to prevent attack, but also to ensure that sensitive data are not released. In this paper, we consider the problem of ensuring that only pre-authorized data leave a network boundary using either overt or covert channels, i.e., preventing exfiltration. We identify the goals of *transparency*, *performance*, and *simplicity*. A system designed to prevent exfiltration should not adversely affect the transfer of authorized data and should work with existing protocols. Key to our approach is: i) separating the process of vetting authorized objects from line-speed data verification; and ii) employing a restricted, but compliant, HTTP subset to limit covert channels. In our evaluation, we show that Glavlit adds little overhead to the operation of a software network bridge.

1 INTRODUCTION

The protection of sensitive electronic data is an increasingly difficult problem. All businesses, governments, and individuals must process sensitive data, and improperly releasing such data can have significant consequences. Consider the inadvertent release of the search queries of hundreds of thousands of AOL users [11]. While organizations must be able to process information not fit for release like intellectual property, financial records, and medical information on their internal network, they must simultaneously process and distribute public data to the outside world.

The goal of our work is to ensure that only approved data exit an organization's protected internal network. We wish to prevent the transmission of data either overtly in the *payload channel* of layer 7 protocols such as HTTP or FTP or in hidden covert channels in the *protocol channel* of these protocols. To prevent information leaks through these channels, everything beyond the TCP header must be verified before allowing each packet to exit the network. Performing such verification at line speed has thus far been limited to pattern searching for sensitive information such as social security or credit card numbers. We, on the other hand, wish to enable verification at the granularity of entire files.

There are known techniques for detecting covert channels in layer 3 and 4 protocols [3, 8, 12]. There is relatively little work in detection at layer 7 despite the existence of many layer 7 channels [4, 6, 9]. We target HTTP for verified data transfer across a network boundary because it is the dominant layer 7 protocol used for interactive data transfer and it is general to a variety of deployment settings. We believe that our techniques extend to other layer 7 protocols, but we leave it to future work to test this hypothesis.

We have developed a system, *Glavlit*[1], which can prevent unauthorized release from a protected network while allowing authorized information to pass unhindered. Our goals for Glavlit are *transparency, performance,* and *simplicity.* Glavlit provides stringent security guarantees by enforcing complex *exit policies* while trusting only the systems responsible for authorizing individual files for release and for inspecting the packets leaving the network. At the same time, the common case performance of Glavlit is comparable to a software network bridge.

We first partition information into *objects*. An object is contiguous related information that can be analyzed independently, such as a file. We split the process of content control on objects into two distinct phases: *vetting* and *verification*. Vetting is the process a designated authority follows to determine whether an object is appropriate for external release. Verification is the process of ensuring an object was previously vetted before releasing it across a designated network boundary. Glavlit verification assumes that *any* machine within the protected network is subject to compromise or negligence.

Many powerful vetting techniques cannot be implemented directly in the network because of performance overheads and the need to operate on entire files rather than individual packets. For example, digital review of complex files such as Microsoft Office formats, PDF, or multimedia can be compute-intensive and potentially require whole-file analysis. Additionally, some organizations are not willing to depend entirely on digital review and require human intervention. One could imagine a new protocol where external users queue requests for particular data. Glavlit allows objects to exit the network

[1] We have named Glavlit for the organization of the former Soviet Union that handled official state censorship matters.

upon verification that they are authorized for release. Unfortunately, this process imposes significant per-request overhead and makes object access highly asynchronous. Hence, we provide a means to *decouple* the vetting process for objects from their verification at a gateway.

To address information leaks in the protocol, we enforce HTTP protocol compliance to detect or limit most covert channels. We perform on-the-fly parsing of the protocol, verify the contents of structured fields, and restrict the HTTP RFC where necessary. We also prevent unstructured channels and timing attacks by correlating request-response pairs and normalizing server response time. In general, eliminating all covert channels for transferring unauthorized data is difficult to impossible. Therefore, we limit the bandwidth of such channels and essentially "raise the bar" for attackers.

The remainder of the paper is organized as follows: We introduce and motivate the Glavlit system architecture in Section 2. Our techniques for preventing leaks are given in Section 3. We briefly evaluate our system's performance in Section 4.

2 GLAVLIT DESIGN

We have designed Glavlit to ensure that *all* information leaving a protected network has been approved for release. In this section we first describe the motivation for creating such a system by examining usage scenarios. We then describe our system architecture and threat model.

Consider the following scenarios: A company X outsources its customer service to another company Y, and they have established a shared network connecting their corporate LANs. X must provide proprietary documents and manuals to Y, but it does not wish to share its customers' personal information or financial records. X can use Glavlit to ensure that private information does not leak from its network to the network shared with Y.

Glavlit could also be used to enforce classified data handling policies among government agencies. Glavlit can dictate that all data that cross an organizational boundary are appropriate for release (e.g., to foreign nationals or to other networks with different classification levels). Glavlit's centralized vetting can also strictly enforce and track the set of granted *exit capabilities*. For example, Glavlit can require that only project leaders may vet files. It can further ensure that each vetting action is fully documented to provide an audit trail.

2.1 System Components

Figure 1 shows the system architecture of Glavlit. A central server called the Warden vets objects. Client software provides an interface to content providers to manage exit policy at the Warden (Step 1). Mechanisms for submitting objects for review and the actual approval process are orthogonal to this effort. The Warden can implement

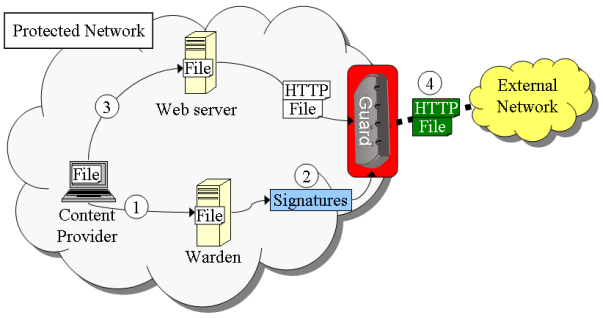


Figure 1: Glavlit System Architecture

any type of digital and/or human reviews to determine if the object is fit for release. The Warden shares a repository of white-listed content allowed to leave the network with the Guard. A *signature* stores hash values and metadata to later verify the content at the Guard (Step 2). After vetting, the user can place the file on a Web server accessible from outside the protected network (Step 3).

The Guard is a high-speed transparent network bridge at the perimeter of the protected network. When an external request arrives for a new object, the Guard parses the request and waits for the server response. The Guard checks verifiable fields in the server response, enforces ordering, and performs timing analysis on the response time to mitigate protocol channels. The Guard determines the boundary between the protocol header and object payload. For every packet containing payload data, the Guard verifies that the (partial) contents match some previously vetted object (Step 4). If verification fails, the Guard can actively stop data from leaving the network by terminating the connection.

To support verification of HTTPS, the Guard can use techniques from intrusion detection systems [2]. Each Web server's private keys can be shared with the Guard, allowing it to perform in-line decryption of all traffic. This sharing is possible because the owner of the Glavlit protected network also controls the internal Web servers.

2.2 Threat Model

We further explore the motivation behind developing this system by examining the threats to information leaks. We categorize these threats into the following:

Accidental Release – In this scenario, a valid system user inadvertently releases information. For example, this could happen because the user did not know a file's content was sensitive and places the it on a public-facing Web server. Since this threat is most common, Glavlit counters it through high-speed content control.

Malicious Use of Standard HTTP – An attacker (e.g., disgruntled employee, external hacker, user inadvertently compromised by virus or malware) compromises a host in the protected network and maliciously places sensitive content on an existing Web server. Since Web servers are

often replicated, allowed through firewalls, and accessed routinely by many parties, it follows that an attacker can use an existing server to deface existing content or use new/existing content to leak sensitive data. For example, Web servers often serve content from shared file systems (e.g., for user home directories) that can be accessed by underprivileged users. Glavlit detects any modification to vetted files and rejects files that are not vetted.

Malicious Use of Another Layer 7 Protocol – An attacker can use a non-HTTP protocol to steal information. We believe that our techniques for preventing leaks extend to other protocols, but must now rely upon other systems and policy to prevent leaks in these protocols. For example, many protected network firewalls only allow certain ports to be accessed externally (by intention implying that only the protocols associated with those ports are allowed). Glavlit provides the additional assurance that another protocol is not being used on an HTTP port.

Compromised Web Server – An attacker has full access to a protected Web server and may modify its configuration or replace it with a rogue server. The attacker can now embed covert channels in HTTP protocol or timing to encode information. Glavlit detects this activity by verifying the validity of all protocol responses and normalizing the server's response time. Even if an attacker creates a rogue server that does not use covert channels, Glavlit only allows it to serve *vetted* content.

Compromised Warden or Guard – Unfortunately, we must prevent this type of attack by assumption. For example, a malicious insider with appropriate permissions can use the Warden to vet unauthorized content, allowing it be released by the Guard. We assume that access to the Warden and Guard is more closely controlled than other hosts in the system like user workstations or Web servers. We also assume that the Warden uses its own *independent* authentication system to grant vetting access.

3 PREVENTING INFORMATION LEAKS

In this section we describe the techniques Glavlit uses to prevent unauthorized content release. Since we must control data release through authorized channels with the same vigor as covert ones, we use two complementary techniques: content control and protocol mitigation.

3.1 Content Control

Since powerful vetting can be time-intensive, we perform content control by splitting vetting from verification. The Guard can then verify at high speed that all content crossing the network boundary is pre-vetted.

There are commercial content control solutions that perform vetting and verification simultaneously on the gateway [7]. The primary drawback to this approach is that it is unable to perform whole-object analysis, critical for supporting modern transfer protocols. Thus, these approaches restrict vetting and verification to pattern matching, e.g., ensuring that individual packets do not contain substrings that resemble social security or credit card numbers. The rate at which this approach can send packets is limited by the speed of its vetting process. Another previous approach is to proxy access to protected content and individually analyze the authorization of each request [5]. Since this requires special client configuration which is not feasible for external clients, it has been restricted to intra-organizational protected networks or for *outgoing* client requests.

3.1.1 Vetting and Static Verification

Users send objects they wish to vet to the Warden, which grants or denies each object the ability to leave the network as specified by policy. This process can be as simple as a keyword search, or as rigorous as requiring approval from a committee of human analysts. Once approved, the Warden partitions the object into *chunks* of 1024 bytes, each hashed with SHA-1. The resulting collection of hashes, named a *signature*, supports high-speed verification at the Guard.

Verification ensures that all information crossing the network boundary was previously vetted. This process consists of locating the data within the network stream and comparing the hash of individual chunks to a pre-existing object signature. To identify an object, the Guard hashes the first 256 bytes of the file content (lookup size). This hash keys the signature table shared with the Warden. We use the Content-Length header to identify files smaller than 256 bytes. Hash collisions are not yet implemented; however, a tree structure within the signature table could implement support for collisions.

Once the Guard identifies the object's signatures, it hashes each chunk of object data and compares the result with the known hash. If any hash value does not match, the connection is bidirectionally terminated by injecting a TCP RESET packet.

To perform verification transparently in the network, the Guard must be able to reconstruct the network communications on-the-fly. Since the Guard performs hashes at a chunk granularity, the packets associated with a chunk can egress as soon as all chunks within a packet are fully verified. Since chunks may cross packet boundaries and since packets may arrive out of order at the gateway, we may have to perform some packet buffering. Content verification is complicated by the presence of the layer 7 protocol in the network byte-stream. We discuss protocol verification in Section 3.2.

The Guard creates a data structure called a *flow* for each TCP connection. As packets arrive, it classifies them by protocol, type, and TCP connection. To sim-

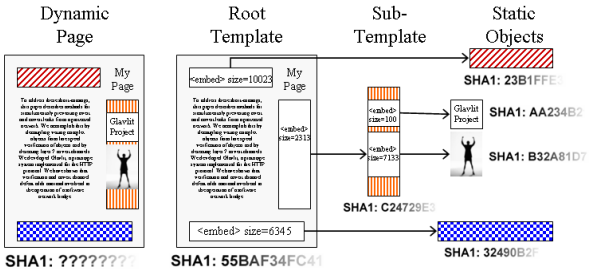


Figure 2: Dynamic Template Model

plify the packet vs. chunk boundary problem, the Guard reconstructs the TCP in-order byte stream in a structure called a *stream*. The Guard maintains an offset to the last verified byte in the stream. As the verifiers operate on the stream, this offset advances until it crosses a packet boundary. The corresponding fully verified packet can be sent out of the protected network at this point.

Since verification may cross packet boundaries, retransmitted packets cannot be re-verified. We handle retransmissions by keeping a window called a *packet cache* of already verified data within the stream. The Guard compares retransmitted packet data with the packet cache and sends it immediately without having to repeat verification.

3.1.2 Dynamic Content Verification

Web traffic is increasingly generated dynamically. The same mechanism used for static content cannot be directly deployed. To solve this problem, we must enhance the vetting and verification process to be aware of the structure of dynamic content. While we believe it is intractable to vet/verify arbitrarily generated dynamic content, we present an approach to handle a subset of dynamically-generated content.

We assume that the dynamic object fits a template model. The template defines static content as well as gaps that contain other templates and static content (Figure 2). The dynamic object can be modeled as a tree where the base template is the root and the leaves are static data. Each layer of the tree defines sub-templates and static content allowed in each gap. Gaps may be filled by another template or a list of valid static objects. For example, a dynamic Web page might randomly choose from a set of images for the header of the page. The web developer must express the template structure and valid objects for each gap as part of the content development [1]. Glavlit vetting and verification processes can now understand this structure because it is self-describing.

We cannot use the Content-Length header to determine the size of embedded objects *smaller* than the lookup size. To solve this problem without an exhaustive search, we propose a solution that requires the Web server to cooperate but does not assume trust using hints. Since the server constructs the content, we require the server to include the size of small dynamic objects within each gap. This does not place trust on the Web server since any false description will misalign the hash.

3.2 Protocol Channel Mitigation

The previous discussion shows how to prevent the transfer of unauthorized data in the payload of HTTP. However, one can transfer unauthorized data within the protocol using covert channels. In this section, we examine the channels for information release using HTTP and what Glavlit does to mitigate them. We discuss the carrier data where information could be encoded covertly. We classify carriers as structured or unstructured [8] and separate them by the degree to which the organization of the data is apparent.

3.2.1 Structured Carriers

Structured data have explicit organization and semantics. We further partition structured carrier channels by their syntactic compliance to the relevant RFC and semantic validity.

We first examine non-compliant channels that deviate from the protocol syntax. For example, a server response could include arbitrary information. Conventional parsing can easily detect this technique. A common case of a non-compliant protocol channel is one that uses the HTTP port to tunnel another protocol.

Since the RFC loosely defines header syntax, adding *Credit-Card: 1234-5678-9012-3456* as a header is actually compliant. The RFC does not explicitly require response headers be in a certain order or to be used at all. An attacker can use list order steganography to encode data by reordering the response headers [13]. In addition to custom headers, the presence or lack of headers could also encode data. Glavlit defines a fixed set of acceptable response headers per request type. The Guard parses outgoing responses using a more restrictive grammar than the one given in the RFC. It also parses incoming requests, but uses the standard RFC grammar to allow any client to connect to a protected Web server.

Responses that do not deviate syntactically are compliant; however, compliance does not imply that the server's response is *correct*. For example, a server may return Content-Length 1024 and 1025 on consecutive requests for the same file of size 1024. Such responses could encode a series of 1's and 0's for covert data delivery. The peak bandwidth in bits per second B of a compliant channel is:

$$B = c * log_2(n) * (t + \frac{RTT}{2})$$

where c = # *connections*, n = # distinct server responses, t = server CPU time, RTT = round trip time

For example, with a 10ms RTT, server processing time of 3ms with 2 connections and 8 distinct server responses, the bandwidth of the covert channel would be 48 bits per second.

In addition to Content-Length, a covert channel can encode data in fields like Last-Modified or Content-Type [4]. Most HTTP header fields are *verifiable* given sufficient information about the nature of the request and the content served. We have examined the HTTP RFC to determine the verifiability of each header. Once we verify a field, only a single response may leave the network, significantly reducing channel capacity. Some header fields are not verifiable because they are based upon server state and load. For example, the Content-Range header specifies the particular chunk of data being transmitted that could vary with respect to the OS buffer cache.

Headers often must be restricted in the server configuration to enable verification. This leads to less flexibility because a header may only take on a few values. In practice, operating a Web server only requires a small number of headers; therefore, we feel this limitation is acceptable. For example, the Allow header specifies the HTTP commands that the server can understand and could be set statically (e.g., only GET/HEAD/PUT methods). Restricted configuration can also be based upon the corresponding request. The Connection header defines the TCP connection state the server will use for the connection. To aid verification, this header can be standardized to always *keep-alive* when an HTTP/1.1 connection is requested and always *close* when HTTP/1.0 or close state is requested.

Our implementation limits the number of distinct responses by verifying protocol fields by combining object meta-data and the corresponding client request to completely verify the server response. The incoming flow data structure stores a queue of parsed requests. When an HTTP response returns, the Guard parses it and compares it with the incoming request and the signature meta-data to ensure that all fields are verified.

3.2.2 Unstructured Carriers

Unstructured carriers store data subjectively or randomly. For example, the order and timing of network events can encode data in an unstructured manner. We examine the *structured artifacts* that these channels produce and ways Glavlit can defeat them.

Request order can encode information unidirectionally *into* a server [6]. Similarly, reordering pipelined responses enables bidirectional communications. The Guard maintains the RFC specification that responses should be returned in the order they were requested by a single client by parsing and correlating incoming requests.

An attacker can encode information based on the timing of network activity [3]. Because of the inherent randomness in network timing, *increases* in the timing carrier (e.g., packet inter-arrival time or HTTP response time) frame covert data. These increases leave a structured artifact behind; therefore, a good channel implementation should affect the timing carrier as little as possible. One can model the Web server performance (e.g., as a function of offered load) and block deviations from the model. This technique has two drawbacks: the model may not be sufficient to cover the server's behavior and the distance metric may not detect carefully crafted deviations.

We can also disrupt rather than detect a covert timing channel by introducing *jitter* into the data stream. This *jitter* is impossible to defeat without increasing the amount of data sent across the network, e.g., a reliable channel, that further reduces usable bandwidth. Kang et al. employed a similar concept to prevent timing channels in the Pump with fully stochastic acknowledgements [10]. We can normalize Web server performance to defeat potential covert timing channels. Glavlit can delay responses by a uniformly random variable or follow a fixed probability distribution. Normalized responses are often not apparent to a user but devastating to a covert timing channel.

The above covert channel mitigation and content control techniques can *limit* the channel capacity. However, there still exist ways to encode information. It is computationally infeasible to completely defeat all covert channels. The only alternative is to limit the number of requests the server is allowed to process per unit time.

4 PERFORMANCE EVALUATION

We evaluated the performance of our prototype Glavlit system compared to a direct wire connection and a standard Linux kernel software bridge (Figure 3). We also tested the Guard with verification *off* to evaluate our network implementation. Our test setup consists of three machines running Linux 2.6.9-34 with dual 2.8 GHz Pentium 4 Xeon processors, 2 GB of memory, and gigabit Ethernet interfaces. The Guard is connected to a host running a custom HTTP client on one interface and a host running Apache version 2.2.2 on the other. The custom client spawns 20 threads, and each thread requests files using HTTP 1.1 for a specified time.

The Guard with verification does not impose significant overhead for most file sizes compared with the kernel bridge or direct connection. The Guard *without* verification performs slightly better than with verification showing that the primary overhead is packet reconstruction and forwarding. The overhead of setting up new connections (protocol parsing and TCP stream allocation) reduces the performance of the Guard with verifi-

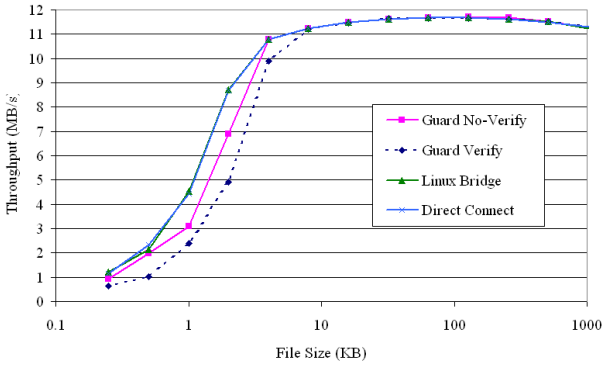


Figure 3: Throughput vs. File Size

cation up to 50% for small files (<8KB). Despite this, the Guard sustained processing approximately 3000 requests per second. For the cases where file size is greater than 10KB, the Guard can hash and forward packets as fast as the server can produce them. In these cases, the per-connection overhead is amortized over larger data transfers. Given this performance with a largely un-tuned implementation, we believe it possible to perform verification at higher speeds with a highly tuned kernel-level software implementation or with hardware acceleration.

5 Conclusion

This paper presents a network content protection system that avoids many of the drawbacks with previous attempts to ensure that only authorized data cross a network boundary. Specifically, the system maintains client-server transparency, while only marginally decreasing throughput. Additionally, our system can secure all files on a network regardless of the security at a particular host.

The key insight behind our approach is the decoupling of the object-vetting process (which can be variably slow) and the object-verification process (which can be performed at high speed and on a per-packet basis). Our prototype implementation of Glavlit performs nearly as well as a standard software bridge.

Malicious users often employ covert channels to transport data outside the network boundary. As an overture to this remaining exit channel, we have implemented the first system to transparently mitigate application-layer covert channels in HTTP.

References

[1] Brabrand, C., Møller, A., and Schwartzbach, M. I. Static Validation of Dynamically Generated HTML. In *Proc. ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering, PASTE '01* (June 2001), pp. 221–231.

[2] Breach*View* SSL. Breach Security, Inc. www.breach.com/products_breachviewssl.asp.

[3] Cabuk, S., Brodley, C. E., and Shields, C. IP Covert Timing Channels: Design and Detection. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security* (New York, NY, USA, 2004), ACM Press, pp. 178–187.

[4] Dyatlov, A., and Castro, S. Tunneling and Covert Channels over the HTTP Protocol, June 2003. http://www.gray-world.net/projects/papers/covert_paper.txt.

[5] Entelligence™Content Control Server. Entrust. www.entrust.com/content-control.

[6] Feamster, N., Balazinska, M., Harfst, G., Balakrishnan, H., and Karger, D. Infranet: Circumventing Web Censorship and Surveillance. In *Proceedings of the 11th USENIX Security Symposium* (Berkeley, CA, USA, 2002), USENIX Association, pp. 247–262.

[7] XPS™Extrusion Prevention System. Fidelis Security Systems. www.fidelissecurity.com.

[8] Fisk, G., Fisk, M., Papadopoulos, C., and Neil, J. Eliminating Steganography in Internet Traffic with Active Wardens. In *IH '02: Revised Papers from the 5th International Workshop on Information Hiding* (London, UK, 2003), Springer-Verlag, pp. 18–35.

[9] Handel, T. G., and Maxwell T. Sandford, I. Hiding Data in the OSI Network Model. In *Proceedings of the First International Workshop on Information Hiding* (London, UK, 1996), Springer-Verlag, pp. 23–38.

[10] Kang, M. H., Moskowitz, I. S., and Chincheck, S. The Pump: A Decade of Covert Fun. In *ACSAC* (2005), pp. 352–360.

[11] Naraine, R. AOL 'Screw-up' Causes Search Data Spill. EWeek, August 2006. www.eweek.com/article2/0,1895,2000225,00.asp.

[12] Tumoian, E., and Anikeev, M. Network Based Detection of Passive Covert Channels in TCP/IP. In *LCN* (2005), pp. 802–809.

[13] Wayner, P. *Disappearing Cryptography: Information Hiding: Steganography and Watermarking (2nd Edition)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.