GreenApps: A Platform For Cellular Edge Applications

Talal Ahmad
NYU
ahmad@cs.nyu.edu

Edwin Reed-Sanchez
NYU
elr210@nyu.edu

Fatima Zarinni
NYU
fzarinni@cs.nyu.edu

Alfred Afutu
NYU Abu Dhabi
alfred.afutu@nyu.edu

Kessir Adjaho
NYU Abu Dhabi
mka325@nyu.edu

Yaw Nyarko
NYU
yaw.nyarko@nyu.edu

Lakshminarayanan Subramanian
NYU
lakshmi@cs.nyu.edu

ABSTRACT

This paper presents the design, implementation and deployment of *GreenApps*, a ground-up platform that enables off-grid, near off-line and highly available cellular applications in rural contexts. The GreenApps platform enables rapid development and deployment of almost-always available applications that can be executed locally on top of open-sourced cellular base stations. The GreenApps platform has been deployed in two rural regions in Kumawu, Ghana and Pearl Lagoon, Nicaragua and supports different community-centric applications.

ACM Reference Format:
Talal Ahmad, Edwin Reed-Sanchez, Fatima Zarinni, Alfred Afutu, Kessir Adjaho, Yaw Nyarko, and Lakshminarayanan Subramanian. 2018. GreenApps: A Platform For Cellular Edge Applications. In *COMPASS '18: ACM SIGCAS Conference on Computing and Sustainable Societies (COMPASS), June 20–22, 2018, Menlo Park and San Jose, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3209811.3212704

1 INTRODUCTION

Despite significant advances in the penetration of cellular networks in developing regions in the past decade, the growth has been predominantly centered in urban regions, with relatively low rural presence [11–13, 15, 16, 18]. The conventional cellular connectivity model has not proven to be economically viable for rural settings due to lack of stable and reliable grid power [9] and due to lack of reliable backhauls and data services [10, 14]. The situation is further exacerbated by the fact that most rural areas have low population densities which translate to lesser revenue for commercial cellular providers.

This paper presents the design, implementation and deployment of *GreenApps*, a platform that leverages programmable base stations to deliver off-grid, near off-line mobile applications that empower rural communities. Recent advances in hardware [1, 2, 6, 8] and open-source software [4, 5] provide a unique opportunity to deploy programmable cellular networks in rural contexts; however, most deployments [7, 13, 19] of such base stations have focused primarily on providing basic communication services like calls and SMS. Heimerl et al. [13] demonstrated the viability of independently run, locally operated cellular networks. Similarly, Rhizomatica [7] has deployed tens of community-run GSM cellular networks in Oaxaca, Mexico with a short-term experimental spectrum license. Zheleva et al. [19] deployed Kwiizya, a system for enabling basic calls and SMS-based service in Zambia. However, there has been very limited work on leveraging these programmable base stations to enable new community-based mobile applications at the edge.

We present our experiences in designing, implementing and deploying GreenApps in hard to reach rural regions in Ghana and Nicaragua. We provide technical details of how we built our platform, as well as details of the development process for our applications within GreenApps. We have deployed our complete end-to-end platform, including applications, in rural regions, in both countries. We also present an evaluation of key components of the GreenApps platform. GreenApps is currently serving populations in Nicaragua and Ghana, and we have preliminary evidence on user satisfaction with our platform. A wide-scale and in-depth study of the impact of our platform on the livelihood of locals in these regions is part of our future work.

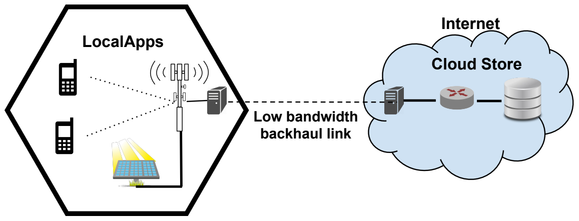


Figure 1: GreenApps System Architecture. GreenApps architecture is described in detail in section 2

2 GREENAPPS ARCHITECTURE

Our GreenApps platform is shown in Figure 1. GreenApps consists of the following 4 main building blocks: (1) an inexpensive open-source GSM base station; (2) a local server; (3) new mobile applications that we built for specific communities; and (4) an Internet backhaul link, whenever available, for opportunistically synchronizing local apps with the cloud store. The main goal of GreenApps is to satisfy the essential needs of local communities via mobile applications that can operate under extreme rural conditions with no power, and no cellular or Internet connectivity. Under the condition where an Internet backhaul link is available, GreenApps exploits this opportunity to provide services distributed across multiple cellular base stations.

2.1 Enabling Edge Application Support

GreenApps supports SMS, voice and data-centric applications locally using an approach which is completely different from how applications are built in conventional cellular networks. In the conventional cellular architecture, messages are routed from base stations to the Internet-facing edge in the core network (after traversing various entities in the core). The Internet-facing edge then routes messages to specific data centers where applications are hosted. While this architecture is designed to be consistent and scalable, it requires reliable links from the base station to the Internet-facing edge in cellular core network. Contrary to this our goals are to reduce dependence on backhaul and to simultaneously provide application support for applications to run locally at the base station. To achieve this, we need to route messages in a scalable manner and prevent any unintentional blocking of critical user communication in this process. We introduce hooks in the routing software at the base station in order to route messages to applications based on the destination short code and send them to the appropriate application engines.

Our platform leverages the hooks and already available APIs to build useful applications in base stations that use the OpenBTS or Osmocom stacks. For each application, the local server coupled with the base-station runs the applications locally and the messages from clients are forwarded by the base station to the correct application. To route an SMS message to a particular application based on the application short code, we used different tools for the OpenBTS and Osmocom software stacks. The OpenBTS stack usually works with `smqueue` to route all messages. Instead we routed the messages to Freeswitch `mod_sms` module specially designed to handle SMS traffic. `mod_sms` enables us to route the messages using a chat-plan defined through XML to route SMS based on the destination phone number. In the Osmocom stack, the `RCCN` library is used to route messages in conjunction with osmobsc. All received messages are routed to a local python server which routes appropriate responses back into the network using the same library. We modified the code of `sms.py` in `RCCN` to route the messages intended for applications (based on the destination short code) to our local application server. Given the current operational constraints of the OpenBTS and Osmocom stacks, we believe a single compute server suffices for running several applications without any performance penalties.

Across both the stacks, we leverage the Asterisk PBX setup for enabling IVR server applications. The Asterisk software is installed locally and all calls to the IVR application short code are bridged to Asterisk. The users can listen to a menu of options and navigate by pressing various keys. Enabling this feature is a standard functionality available in Freeswitch and any other PBX software.

2.2 Identities and Distributed Applications

A key building block for supporting different forms of applications in GreenApps is an identity management layer. GreenApps supports three basic types of identities: network identities, user identities and application identities. In GreenApps, the common user identity is the the International Mobile Subscriber Identity (IMSI) which is physically written on a SIM card. The network identity is the identity issued by the network against the IMSI. This is typically the phone number of the user, in our case this refers to a local phone number identifiable within the base station.

In GreenApps, distributed applications can be built by using application specific identities, which refer to unique information state of the application that may be tied with one or a group of user identities. The identity space corresponding to an application can be viewed as a hash table or a key value store. For example, consider the case where two users wish to perform a BUY/SELL transaction in GreenApps across multiple cellular sites connected via a server in cloud; each item transacted is associated with a unique application identity and each buy/sell bid is a mapping between an application identity and a set of user identities. The complete key/value store of such identities in an application forms the application state. The application state is manipulated by simple put/get operations when the application is running locally, but if an application is running in a distributed manner across multiple base stations, then the local state across each base station needs to be selectively synchronized with the cloud state. To achieve this, we provide primitives for distributed applications running across multiple base stations.

2.3 Syncronization Primitives

In GreenApps we provide some primitives for synchronization between cellular nodes and a cloud based server. These primitives are available to all applications using a python library. These primitives provide a flexible and convenient way to upload data to the server and enable distributed applications when the backhaul is available. Each primitive gives the application a choice of syncronization in a slightly different way based on urgency of syncronization for a particular application funcationality.

`1. SLOWPUT(Identity i, Type t, Data d):`

GreenApps is designed to provide local applications but it is designed to provide synchronization with the cloud if there is a need. To this end, for each application identity the application logic is provided a method to define synchronization requirement. The Internet link between the base station and the cloud has a low bandwidth-delay ratio and is typically intermittent. This raises questions regarding what identities need to be synchronized to ensure high availability to a user while having consistency in the application state at the cloud. First, to ensure high availability of identity lookup, GreenApps supports quick local authentication of user identities; pre-registered users and two users within the same base station can communicate with each other without the need for backhaul connectivity to the cloud. Second, we provide synchronization support for each of the application identities so that users can pick and choose the consistency over availability where necessary by ensuring the data is saved on the cloud before it is saved locally.

In the event, the GreenApps base station has a backhaul link with moderate bandwidth, GreenApps can enable lazy synchronization of application state between the local server instance and a cloud instance of the same application to enhance end-to-end reliability. GreenApps supports an application agnostic primitive *SLOWPUT*, which is a bandwidth-aware mechanism to synchronize data between the local node hosting applications and a cloud data store. SLOWPUT provides a controllable way to synchronize application state to the cloud. We achieve this by maintaining a common queue across various applications. This queue contains different data items(from different applicatioins) that need to be uploaded to the cloud and uses a fair queuing mechanism to allocate the bandwidth between competing requests.

We used Memcached [3] to provide the functionality of lazy synchronization. Memcached is a high-performance, distributed memory object caching system which can be used for general applications. A queue is maintained using a linked list of different objects in cache. An item is added to the queue if an application executes the SLOWPUT primitive. There is a separate program that is always running and keeps uploading the items in the queue if the bandwidth is available. The items are uploaded on a first come, first served basis. The function call for SLOWPUT is as follows: `SLOWPUT(Identity i, Type t, Data d)`. In this call, user `i` communicates marshalled data `d` for application `t`.
2. `FASTGET(Identity i, Type t, Data d)`:

This primitive is used by the local application to send and receive data from the server in an *active* manner. The parameters include a user `i` communicating data `d` for application `t`. This primitive is usually called as a result of an application function when there is a higher urgency needed for consistency in application state. The communication is not queued, as in `SLOWPUT` primitive. Instead, this primitive is executed instantaneously, provided the the backhaul link is available. For example, any inconsistencies in the case of a banking transaction are grave, and a strong consistency has to be ensured even if the user does not get a response immediately. Such applications use the `FASTGET` primitive and marshall the transaction in the data parameter, forwarding it to the central server of the banking application.

3 APPLICATIONS

In this section, we describe three applications we deployed using GreenApps in Ghana and Nicaragua. These applications are used for various useful functions in the communities like dissemination of information, entertainment and communication.

3.1 Fishline

Fishline is an SMS-based application that enables small business owners to send out mass advertisements over SMS to people in the community. This application has been built and is running in the Pearl Lagoon GreenApps platform. To motivate the need for Fishline we conducted 6 qualitative interviews in Pearl Lagoon. People interviewed included 2 artisan fisherman, 2 deep water fishermen, and 2 owners of fishing cooperatives. We chose various poeple associated with the fishing community because that is how most people earn a livelihood in the area. This cross section of fishermen represents the 3 basic types of individuals working in the local fishing economy. We asked them questions about demographics, their fishing practices, and the economics of Pearl Lagoon. All the interviewees suggested interest in the idea of an application to advertise fish for sale. Fishing cooperatives also saw a use for the Fishline, since they would be able to get messages earlier before fishermen are on shore, and can prepare for large catches as they arrive. Fishline is an entirely local application and no syncronization primitives were used in the Fishline application.

3.2 P2P Transactions Marketplace in Ghana

We built a P2P transactions application that uses an SMS-based interface to enable traders to sell comodities to customers in the community. We designed our application in a way where farmers and traders could look up the prices that others are offering for the product before deciding whether they want to sell or buy from a particular user. There are three kinds of messages sent, SELL, BUY and SEARCH. The SELL message is of the format,"SELL <commodity> <amount> <price>". For example, if farmer wants to sell 5kg of corn for 10 cedi[1], then, the message "SELL corn 5kg 10" is sent via SMS, and this message is saved in a database along with the user's number who sent it. Subsequent people who are interested in buying or looking up the price can do so using a BUY or SEARCH SMS. This application was deployed in our Ghana setup.

Whenever the messages match a short code for P2P application, the message is routed to the HTTP server hosting the application code. The lazy `SLOWPUT` is used to send data to the server on sale of a commodity. Therefore when a user tries to sell, we acknowledge the sell has been received locally(by sending a message back to seller) and transfer the sell to the cloud in a lazy manner. In case the user is interested in purchasing a commodity, `FASTGET` is used, and the data returned is sent to the user over one or more text messages. This is because we want strong consistency and do not want to give user a response immediately without ensuring that we are actually able to update the record in the cloud.

3.3 IVR-Based Social Media

Traditional IVR systems have required access to either a cloud based service or have relied on a local server with a cellular dongle connecting to a local cellular network service. The key benefit of the GreenApps platform powered IVR application is to enable an off-line and off-grid IVR application. By attaching the server functionality to the base station, the application server functionally locally resides on the programmable base-station without the need for backhaul connectivity.

The key building block for this application is the local code that glues the IVR to the cellular system. The important function is the `ReceiveHandler` running at local server, which is invoked using a HTTP request on entering the IVR extension. After the initial HTTP request, the call is bridged from base station to Asterisk IVR server running locally. The Asterisk server logs each input of user as entered. If the user is recording a message, the `ReceiveHandler`

[1]cedi is the local currency in Ghana

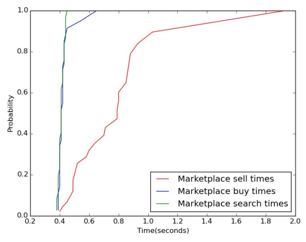
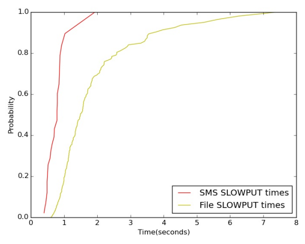
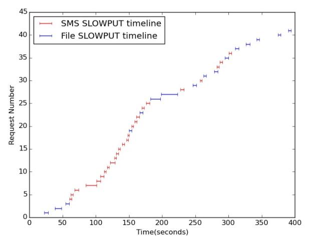


Figure 2: CDF of time it takes for the different functions in the P2P Marketplace application



(a)



(b)

Figure 3: CDF of time taken for the `SLOWPUT` of file vs SMS and a view of the queue in the P2P Marketplace application

is executed after its completion. The `ReceiveHandler` uses the lazy `SLOWPUT` primitive to send the recorded file to the cloud component for lazy syncronization.

4 APPLICATION PERFORMANCE AND EXPERIENCES

We briefly outline some of the early implementation and deployment experiences in running these applications.

Marketplace Application: As a basic setup we had two base stations, one in Ghana and the other one in New York. We developed an Android application that enables phones to automatically `SELL` and `BUY` items in a periodic manner. We had 4 phones sending `SELL` messages and 3 phones sending `BUY` and `SEARCH` messages. 5 of the 7 phones were connected to the BTS in Ghana and the others were connected to the base station in New York. Simultaneous with the above application we made calls manually to the IVR application and ran a script at local server to generate data. This helped us simulate the local component receiving data from IVR and sensing applications.

We report the performance of the P2P transactions application in Figure 2. The graph above shows the end-to-end latency between the local application and server for the `BUY`,`SELL` and `SEARCH` operations. It is important to recall that the `SELL` operation translates to the `SLOWPUT` primitive, while `BUY` and `SEARCH` translate to `FASTGET` primitives. It can be observed in Figure 2, there is a clear distinction between the time it takes for `SELL`, compared to `BUY` and `SEARCH`. This is expected because `SELL` requests are queued and processed in a lazy manner while the BUY and SEARCH are executed using the urgent FASTGET primitive. It is important to note that no failures were experienced and all messages received at base stations were transferred to the server. At a user level user got SMS back for all the functions they performed. They got SMS for `SELL` immediately because the local component of the application gives the response immediately and then executes the `SLOWPUT` operation while the response for `BUY` operation is sent only after response from cloud to maintain consistency.

IVR Application: File upload is a common operation of the voice-based applications. Figure 3a shows the CDF of the time it takes for different `SLOWPUT` requests on the New York-based node. Figure 3b shows the queue and how requests are processed for the `SLOWPUT` primitive for the base station in Ghana. Figure 3 shows that SMS-based applications will work faster than the file-based applications as SMS-based messages are smaller than files. It also shows that file-based applications can slow down the SMS-based applications. This is true as both are placed in a common queue for processing, and servicing larger files is more time consuming than servicing smaller SMSs.

Early User Experiences: The Fishline application is being currently used in Nicaragua by various users to advertise through SMS in the local community. Several advertisements have been sent out using the service and many users have responded back to most advertisements. More businesses and fishermen are showing interest in enrolling for this service. As a simple example, when we sent out this broadcast message: "messages sent on 30000 are part of a new service pearl cel wants to offer to businesses and institutions to publicize info. If interested reply 'yes RP' ", we got several positive responses from interested users including SMS responses like: "How could I do that?" . Performing a full fledged impact study of the outcomes of the Fishline application on the community is part of our future work. The P2P marketplace application has been tested with cohorts of farmers and traders from the local community to perform BUY/SELL transactions. The IVR-based social media application was inspired by the success of voice-based citizen journalism platforms like Polly [17] and shows how one can build such voice-based applications to function effectively in rural contexts in an intermittency-aware manner.

5 ACKNOWLEDGEMENTS

We thank Zohaib Jabbar, Fareeha Amjad, Tiffany Tong, Yasir Zaki and other members of the Center for Technology and Economic Development (CTED) in Abu Dhabi and CTED Kumawu members in Ghana for their help in this work. We also thank Leslie Greenguard, David Mordecai, Samantha Kappagoda and Thomas Pruett for their support and encouragement. We also thank the anonymous reviewers for their reviews which helped in improving this paper. This work was partly supported by the NYU Abu Dhabi Research Institute, the Center for Technology and Economic Development (CTED) in NYU Abu Dhabi and a CTED-Kumawu Research Grant in NYU Ghana. This work was also partly supported by a Cisco research award.

REFERENCES

[1] 2017. Facebook OpenCellular. (2017). https://code.facebook.com/posts/1754757044806180/introducing-opencellular-an-open-source-wireless-access-platform
[2] 2017. GSM LiteCel by Nuran Wireless. (2017). http://nuranwireless.com/products/gsm-litecell/

[3] 2017. Memcached. (2017). https://memcached.org/

[4] 2017. OpenBTS. (2017). www://www.openbts.org/

[5] 2017. Osmocom. (2017). https://osmocom.org/

[6] 2017. Range Networks. (2017). http://www.rangenetworks.com/

[7] 2017. RHizomatica Community Base Station. (2017). http://rhizomatica.org/

[8] 2017. Sysmocom. (2017). https://www.sysmocom.de/

[9] Talal Ahmad, Shankar Kalyanaraman, Fareeha Amjad, and Lakshmi Subramanian. 2015. Solar vs diesel: where to draw the line for cell towers?. In *Proceedings of the Seventh International Conference on Information and Communication Technologies and Development*. ACM, 7.

[10] Aditya Dhananjay, Ashlesh Sharma, Michael Paik, Jay Chen, Trishank Karthik Kuppusamy, Jinyang Li, and Lakshminarayanan Subramanian. 2010. Hermes: data transmission over unknown voice channels. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*. ACM, 113–124.

[11] Vijay Gabale, Ashish Chiplunkar, Bhaskaran Raman, and Partha Dutta. 2011. DelayCheck: Scheduling Voice Over Multi-hop Multi-channel Wireless Mesh Networks. In *COMSNETS*.

[12] Vijay Gabale, Bhaskaran Raman, Kameswari Chebrolu, and Purushottam Kulkarni. 2010. LiT MAC: Addressing the Challenges of Effective Voice Communication in a Low Cost, Low Power Wireless Mesh Network. In *Proceedings of the First ACM Symposium on Computing for Development (ACM DEV '10)*. ACM, New York, NY, USA, Article 5, 11 pages. DOI:https://doi.org/10.1145/1926180.1926187

[13] Kurtis Heimerl, Shaddi Hasan, Kashif Ali, Eric Brewer, and Tapan Parikh. 2013. Local, sustainable, small-scale cellular networks. In *Proceedings of the Sixth International Conference on Information and Communication Technologies and Development: Full Papers-Volume 1*. ACM, 2–12.

[14] Michael Paik, Ashlesh Sharma, Arthur Meacham, Giulio Quarta, Philip Smith, John Trahanas, Brian Levine, Mary Ann Hopkins, Barbara Rapchak, and Lakshminarayanan Subramanian. 2009. The case for SmartTrack. In *Information and Communication Technologies and Development (ICTD), 2009 International Conference on*. IEEE, 458–467.

[15] Rabin Patra, Sergiu Nedevschi, Sonesh Surana, Anmol Sheth, Lakshminarayanan Subramanian, and Eric Brewer. 2007. WiLDNet: Design and Implementation of High Performance WiFi Based Long Distance Networks. *Proceedings of NSDI 2007* (2007).

[16] Bhaskaran Raman and Kameswari Chebrolu. 2005. Design and Evaluation of a new MAC Protocol for Long-Distance 802.11 Mesh Networks. In *ACM MOBICOM*.

[17] Agha Ali Raza, Mansoor Pervaiz, Christina Milo, Samia Razaq, Guy Alster, Jahanzeb Sherwani, Umar Saif, and Roni Rosenfeld. 2012. Viral entertainment as a vehicle for disseminating speech-based services to low-literate users. In *Proceedings of the Fifth International Conference on Information and Communication Technologies and Development*. ACM, 350–359.

[18] Sonesh Surana, Rabin Patra, Sergiu Nedevschi, Manuel Ramos, Lakshminarayanan Subramanian, Yahel Ben-David, and Eric Brewer. 2008. Beyond Pilots: Keeping Rural Wireless Networks Alive. *Proceedings of NSDI 2008* (2008).

[19] Mariya Zheleva, Arghyadip Paul, David L. Johnson, and Elizabeth Belding. 2013. Kwiizya: Local Cellular Network Services in Remote Areas. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '13)*. ACM, New York, NY, USA, 417–430. DOI:https://doi.org/10.1145/2462456.2464458