# Optimizing Mobile Application Communication for Challenged Network Environments

Waylon Brunette[1], Morgan Vigil[2], Fahad Pervaiz[1], Shahar Levari[1], Gaetano Borriello[1], and Richard Anderson[1]

[1] Department of Computer Science and Engineering, University of Washington, Seattle, WA,
{wrb,fahadp,levaris,gaetano,anderson}@cse.uw.edu

[2] Department of Computer Science, University of California Santa Barbara, Santa Barbara, CA,
mvigil@cs.ucsb.edu

## ABSTRACT

Designing mobile applications for challenged network environments necessitates new abstractions that target deployment architects, non-developers who are charged with adapting an ensemble of off-the-shelf software to a deployment context. Data transfer is integral to mobile application design and deployments have inherent and contextual requirements that determine what data should be transferred and when. In this paper, we investigate building mobile applications in challenged network environments by focusing on abstractions to support disconnected environments and areas of sparse heterogeneous connectivity. We explore and characterize various methods of transmitting data using: existing synchronization tools, peer-to-peer communication, and sparse networks. We also introduce a new software tool called ODK Submit to help streamline application customization to challenged network environments.

## Keywords

mobile devices; Open Data Kit; application framework; multiple-channel communication; peer-to-peer networking; multi-network

## 1. INTRODUCTION

Network connectivity is a persistent concern for organizations working in resource-constrained contexts because 1) connectivity is not always present, 2) the type of connectivity often varies by location, 3) data transmission costs may be too high for limited budgets, and 4) administrative protocols restrict how data can be transmitted and stored. These constraints can inhibit a mobile application's capabilities, implying that deploying organizations need to adapt their application to factor in connectivity and other contextual limitations. Various ICTD research projects have focused on improving connectivity by extending communication infrastructures (e.g., long distance WiFi, village base station, mesh networks). However, customizable software frameworks could help improve mobile applications' capabilities in challenged network environments until universal connectivity is available and affordable everywhere.

Mobile devices often have several built-in transmission capabilities (e.g., GSM, WiFi, peer-to-peer) but lack a flexible framework to systematically adjust to changing network conditions based on an application's deployment requirements instead of simple connectivity available recognition. Data transfer is integral to an application's usage and context making it difficult to create a universal solution to address diverse requirements. This paper argues for creating a software tool that selects appropriate data for transmission over available network channels and can be customized by deployment architects. Deployment architects are generally non-programmers who adapt an ensemble of off-the-shelf software to a deployment context. Providing flexible transmission management to deployment architects could improve the feasibility of deploying mobile information systems in challenged network environments by enabling application-level communication optimizations.

Challenges for deploying applications in developing regions have been documented [10] and include: low literacy, limited technical personnel, use of inexpensive multipurpose devices, and context-specific customization. Research initiatives that address some of these challenges have focused on interface design [16, 28] and rapid customizability [19] to produce frameworks such as Open Data Kit (ODK) [11, 19] and CommCare [17]. These frameworks focus on lowering technical barriers to assist organizations in deploying information services in resource-challenged contexts and are designed for disconnected operation. However, ODK leaves decisions about when and how to transmit data to the end-user which leads to possible inefficiencies with respect to transmission costs or deadlines.

In this paper, we examine options to enable non-developers to adapt their mobile application to various network conditions. To motivate the need for a configurable data transmission tool that operates in sparse connectivity, we discuss challenges with existing paradigms, characterize the performance of popular data transfer apps from different locations, and formalize sources of transmission meta-data into three perspectives. We then propose an (ODK) extension called ODK Submit that uses an organization's deployment parameters to guide communication decisions. Submit enables application-level communication optimization of sparse heterogeneous networks by sending appropriate data over available network infrastructure or peer-to-peer communications. We also investigate and characterize Android peer-to-peer transfer methods to better understand deployment trade-offs for different use cases.

*ACM DEV 2015,* December 1–2, 2015, London, United Kingdom.
© 2015 ACM. ISBN 978-1-4503-3490-7/15/12 ...$15.00.
DOI: http://dx.doi.org/10.1145/2830629.2830644.

## 2. CHALLENGES BUILDING MOBILE APPS

Building and deploying mobile data collection and decision support applications can be challenging, particularly because the task of bridging the design-reality gap [20] is left to the non-programmer deployment architect. In this section, we outline some of these challenges and discuss assumptions that inhibit deployment customization across diversely challenged networking environments.

### 2.1 Existing Paradigms

Challenges with deployments are often magnified in resource-constrained environments because of insufficient design paradigms.

**Uniform Data:** Existing routing paradigms often assume that inherent data properties are sufficient to determine the appropriate network technology for data transmission [15, 24]. This assumption overlooks the fact that data is not uniform but instead has two distinctive qualities: inherent qualities *and* contextual qualities. Inherent qualities of data, such as data types and sizes, are independent of the application in use. In contrast, contextual data qualities are necessarily dependent on use scenarios. Examples include data priority, data importance, deadlines, and precedence. Contextual qualities such as an organization's data policy and local laws can also affect how data is stored and transmitted (e.g., private medical records vs public data).

**Single-Task Mobile Apps:** Resource-constrained environments often lack enough technical personnel to build and customize information systems. This leads organizations to use productivity software (e.g., MS Excel, MS Word) to create solutions that can be customized by staff having little programming expertise. These tools have been designed for conventional PCs that are poorly suited to these limited infrastructure environments. Although mobile devices are well-suited to scarce connectivity and sporadic grid power, mobile software tools do not yet offer the same range of features as customizable PC productivity tools. Instead, several small apps focused on single tasks are created leading to specialized apps with minimal customizability. Mobile frameworks, such as ODK 2.0 [11], are needed to help organizations customize and refine their apps to their context while maintaining the single-task paradigm. Additionally, with single-task apps there is often limited coordination of system resources making it challenging to conserve resources. For example multiple apps could simultaneously attempt to communicate when connectivity becomes available.

**Similar Transmission Cost:** Developers often choose a single transport protocol such as TCP/IP or SMS because of systems abstractions and availability of networks for the original deployment location. However, the cost associated with connectivity can vary across different regions creating feasibility issues for deploying applications in varying contexts. For example, a 500MB post-paid mobile broadband subscription in Europe costs 1% of per capita GNI. By contrast, the same subscription costs 38% of the average per capita GNI across Africa [6]. Even as the cost of broadband subscriptions falls globally, an entry-level broadband connection continues to cost over 100% of per capita GNI in less developed countries, as compared to only 1% of per capita GNI in more developed countries [3]. Even in developed regions, there are communities yet to be covered. In the US, broadband coverage on Native American reservations is less than 10% per capita [5], despite coverage of over 70% for the rest of the country. Although technologies with universal connectivity options like satellite uplinks exist, financially constrained organizations cannot afford them. Restricting data transmission to a single protocol can lead to missed opportunities in optimizing transmission costs based on contextual qualities of data.

### 2.2 Example Usage Scenarios

Based on our deployment experiences, we outline scenarios that highlight networking challenges and the benefits of leveraging contextual data properties when making data transmission decisions.

**Scenario 1 - Site Visits:** The Government of Punjab's Health Department (Pakistan) used ODK to document the workload, staff attendance, and available medical supplies at health clinics. To identify shortages it dispatched inspectors to verify inventory and photograph workers for attendance verification. This information was transmitted to ODK servers; however, only the medical supply data was time-sensitive. The photographic verification of attendance was a human resource concern and did not require urgent delivery, yet the non-urgent piece of data dominated the transmission cost because the size of the photos are large.

**Scenario 2 - Forest Monitoring:** The Surui, an indigenous Amazonian tribe, uses ODK to inventory their forests for selling carbon credits in voluntary carbon markets to create a tribal income stream and protect their environment. Workers hike or boat into remote areas with little connectivity and spend days or weeks conducting carbon inventories. Workers inventory a section of the forest to determine the amount of carbon stored in the trees and upload this data on their return. Data loss and/or corruption can be expensive since expeditions are time-consuming. A simple solution to combat data loss is to replicate data in the field across multiple devices. When workers are in the field they also record signs of illegal logging activity. In contrast to inventory data, reports of illegal activity should be quickly transmitted to authorities via any available connectivity (possibly expensive) to increase the likelihood of catching the perpetrators.

**Scenario 3 - Community Health:** Community Healthcare Workers (CHW) frequently travel to villages to provide basic care. Their visits at households or low-end clinics provide an opportunity to document patients, perform a quick analysis of whether a patient requires referral for further attention, and educate people about best-practices[16, 26]. For example, AMPATH's home-based HIV/AIDS counseling and testing program in Kenya used ODK to reach over a million patients at home. However, CHWs' duties can vary per organization and country thus producing variation in data priority. For example, supervisors may need to receive updates frequently about which tasks have been completed (via SMS or cellular data) but the transfer of patients' medical data may be postponed until a WiFi connection is available. Ministries of Health would likely want an immediate notification on the detection of Ebola or Polio to rapidly quarantine Ebola areas to contain an outbreak or quickly deploy polio vaccination teams to surrounding communities.

### 2.3 Existing Synchronization Tools

Cloud-based data storage and synchronization systems often serve as building blocks for application development. To better understand the performance of existing tools, we evaluated the operation of three popular cloud-based systems: Dropbox [1], OneDrive [2], and Google Drive [3]. We measured performance on Android devices using libraries provided by the cloud services since we are evaluating developer options for creating mobile data applications. Since not all contexts suffer from challenged networks, we chose three cities (Lima, Peru; Kisumu, Kenya; & Lahore, Pakistan)in somewhat resource-constrained countries. Large cities have better infrastructure than their rural counterparts demonstrating best-case scenarios for countries with resource constraints. To provide context for the

[1]https://www.dropbox.com/home
[2]https://onedrive.live.com/
[3]https://www.google.com/drive/

Table 1: Network measurements from various locations

| LOCATION | RTT (ms) | BANDWIDTH (Mbps) | LOSS (%) |
|---|---|---|---|
| Lima | 1173.0 | 1.05 | 0 |
| Lahore | 207.0 | 1.05 | 0 |
| Chowchilla | 154.6 | 1.69 | 0 |
| Pala | 63.0 | 6.93 | 0 |
| Seattle | 39.2 | 11.00 | 0 |

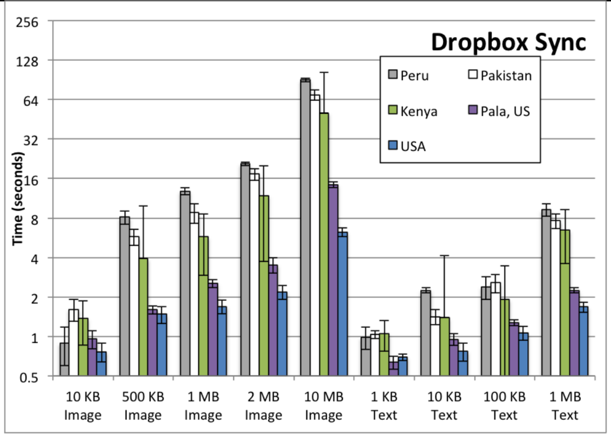


**Figure 1: Dropbox file synchronization performance with varying file sizes using mobile data connection. (Log Scale)**

performance divide between urban and rural areas, we also evaluate the performance differences of an urban city in the U.S. (Seattle, WA) and rural towns in the U.S. (Chowchilla and Pala, CA), which have populations of ~600K, ~18K and ~1.5K respectively. Service carriers used in the experiments include Claro (Lima), Telenor (Lahore), T-Mobile (Seattle), Verizon (Chowchilla) and AT&T (Pala). File synchronization was measured 15 times per service using 10KB, 500KB, 1MB, 2MB, and 10MB of image files and 1KB, 10KB, 100KB, and 1MB of text files.

Table 1 shows network statistics recorded using ping and iperf tools. Cellular networks in Lima and Lahore tend to be highly latent with low bandwidth capacity with Lima experiencing the longest round trip times at 1,173 ms. Seattle was 5 and 30 times faster than Lahore and Lima respectively. Mobile network connectivity in rural US test sites was significantly lower as Chowchilla had 1.69 Mbps (154.6ms RTT) and Pala had 6.93 Mbps (63.02ms RTT) compared to Seattle's 11 Mbps (39.2ms RTT) high speed bandwidth. We emphasize that network availability and performance in Chowchilla and Pala represent a best case for rural connectivity in the US, as measurements were taken from the more densely populated town centers. We also note that poor network infrastructure is not just a problem for developing countries, but for rural parts of developed countries as well.

Results from the performance tests shown in Figure 1, 2 and 3, reveal that Dropbox performs better with all tested file sizes. This is unsurprising as the Dropbox API compresses all files prior to transmission, while Google Drive and OneDrive do not. OneDrive only performed differential synchronization of Microsoft Office files, which excludes many large media files. Google Drive did not use differential synchronization for any file, causing it to use the most bandwidth per file update of the three evaluated options. When accessed via the developer API, Dropbox does not provide differential synchronization, though it does perform data compression prior to transmission. We also note that OneDrive experienced higher variability in file transfer times than Google Drive and Dropbox, with a standard deviation of 5.2 seconds compared to 2.3 seconds for Dropbox and 4.6 seconds for Google Drive. Even though Lahore, Kisumu, and Lima have access to mobile Internet, the performance of the connections were inconsistent and more prone to high latency

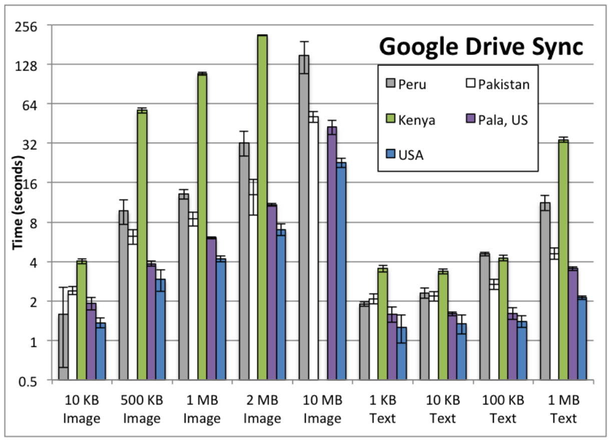


**Figure 2: Google Drive file synchronization performance with varying file sizes using mobile data connection. (Log Scale)**

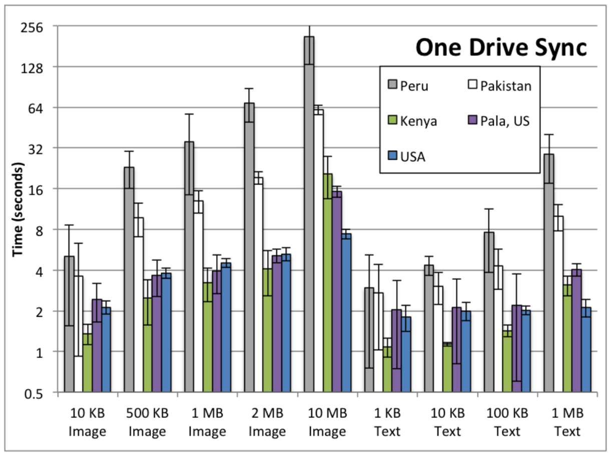


**Figure 3: OneDrive file synchronization performance with varying file sizes using mobile data connection. (Log Scale)**

and limited bandwidth than connectivity in Seattle.

Our measurements demonstrate that common data synchronization platforms experience issues in varying network environments as they all experienced longer file transfer times and greater variability in transfer time in resource-constrained environments. Based on our experiments, Dropbox would be the preferable 'off-the-shelf' solution, followed by Google Drive, then OneDrive. However, there are still issues that these cloud synchronization platforms do not address such as: 1) they lack support to enable organizations to treat data differently based on contextual data qualities; and 2) they are TCP/IP based and do not allow for alternative connectivity options in challenged network environments.

## 3. PERSPECTIVES

Data is often transmitted using whatever protocol a software developer selected when developing the software. Dynamic selection of available protocols based on a deployment's context and user location could improve connectivity in challenged network environments. To better facilitate dynamic selection the traditional concept of the TCP/IP Application Layer [27] should be extended to include: 1) metadata from the platform about connectivity; 2) data properties from the software/application developer; and 3) contextual constraints from an deployment architect. The deployment architect is the domain expert who deploys the software in the field and customizes it to meet the needs of their business or organization. The deployment architect and the software developer both

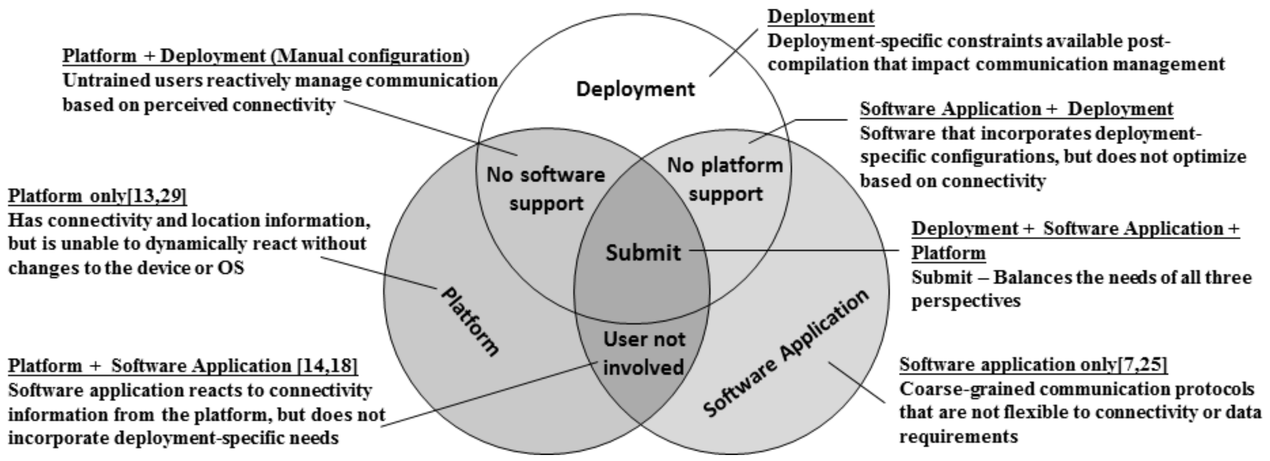

Deployment
Deployment-specific constraints available post-compilation that impact communication management
Platform + Deployment (Manual configuration)
Untrained users reactively manage communication based on perceived connectivity
Deployment
Software Application + Deployment
Software that incorporates deployment-specific configurations, but does not optimize based on connectivity
No software support
No platform support
Platform only[13,29]
Has connectivity and location information, but is unable to dynamically react without changes to the device or OS
Submit
Deployment + Software Application + Platform
Submit – Balances the needs of all three perspectives
Platform
User not involved
Software Application
Platform + Software Application [14,18]
Software application reacts to connectivity information from the platform, but does not incorporate deployment-specific needs
Software application only[7,25]
Coarse-grained communication protocols that are not flexible to connectivity or data requirements

**Figure 4: Design space of communication solutions for utilization of heterogeneous networks**

provide vital information that is necessary to understand an application's communication context and constraints. Our approach of breaking the application layer into parts is similar to Martins et al.'s approach to coordinating different perspectives on system power[22]. We find their multi-perspective approach to optimizing battery life suitable to optimizing communication resources. As Martins et al. point out: *"the user needs to drive"*; claiming: *"1) The OS cannot always know the resource priorities of all applications; 2) applications cannot always know the functionality priorities of the end-user; and 3) users should choose the right level, trading off functionality versus lifetime."* Overall we agree with these insights with the exception of focusing on the end-user. Instead, there is often a deployment architect that handles organization-wide restrictions and imposes constraints derived from deployment considerations. The focus on a deployment architect in addition to the end-user is an important distinction, as grouping developer, deployment architects, and end-users into a single concept can make system optimizations difficult. Figure 4 shows how Submit aims to combine information from the network perspectives of the platform, software developer, and deployment architect to efficiently manage communication resources to relieve the end-user of communication management.

**Platform Perspective -** The platform perspective encompasses the device and operating system perspectives on connectivity and mobility. For instance an Android device can: detect the type of available network connectivity, detect device mobility, estimate available data capacity, and estimate the device's geographical location. Location and mobility information enable Submit to possibly infer the duration of a connection and apply regional data policies. However, the platform is unaware of what policies, financial restrictions, and other data priorities a user or organization may want applied. There are numerous works related to a platform-only communication management scheme as multiple communication channels can be dynamically allocated based on availability or bonded to create compound channels with greater throughput capacity[14, 30]. While Submit dynamically schedules traffic based on channel availability, it does so without modifying the underlying platform to combine or bond channels.

**Application Developer Perspective -** The application developer perspective encompasses issues relating to the functionality of a mobile application. A developer understands the inherent properties such as the type and typical size of data that can help Submit develop an appropriate cost model. Unfortunately, a developer may be biased towards a particular communication medium and may not bother including functionality to support alternatives such as: local off-line storage to support disconnected operation or transmission of summary data over SMS. Thus, developers constrain communication resources via software design and protocol selection. Some examples of developer limitations include apps that communicate over 2G and 2.5G networks exclusively rather than 3G data networks[7, 29]. Likewise, a single protocol limits what an app can effectively communicate, for example, transmitting binary over SMS is non-optimal. Furthermore, a developer likely does not fully understand how a future user may want to deploy the application in varying context with limiting data policies and budgets.

**Application Deployment Perspective -** The deployment perspective encompasses issues relating to contextual deployment requirements that should be incorporated by a deployment architect, as the dynamic contextual information is not available when the developer compiles the software. A deployment's requirements can provide important metadata including information prioritization, and financial restrictions that may change during the lifetime of the project. ODK tools are designed to be general-purpose and have been deployed in a variety of settings including public health, environmental conservation, and census applications. These domains have different needs and real-time information may change how data is transmitted. For example, in a health application, a CHW may find a patient needing immediate referral to a care facility. This message is urgent and should be sent with a different priority than updating a healthy patient's medical record. Usage context is not predictable by the developer nor can the platform impose that one particular channel be used as that channel may not be available. Depending on the urgency of the data it may be necessary to send the same data over multiple channels to ensure delivery or possibly reach multiple destinations. Submit seeks to remove the user burden of actively monitoring the status communication events.

**Overlapping Perspectives -** There are points where each of these perspectives overlap and interact. The most commonly combined perspectives are those of the platform and software application. Opportunistic off-loading approaches combine the connectivity awareness of the platform with software application protocol decisions [9, 18, 21]. While this approach provides more guidance than either platform or software developer perspectives alone, it results in coarse-grained communication automation. In contrast, web applications exemplify the absence of the platform perspective [4, 1]. While there is value to platform-independent systems, platform information about connectivity and location is necessary for maintaining historical connectivity models. Submit's design incorporates information from the different perspectives and attempts to hone how and when information is transmitted in challenged networking environments.

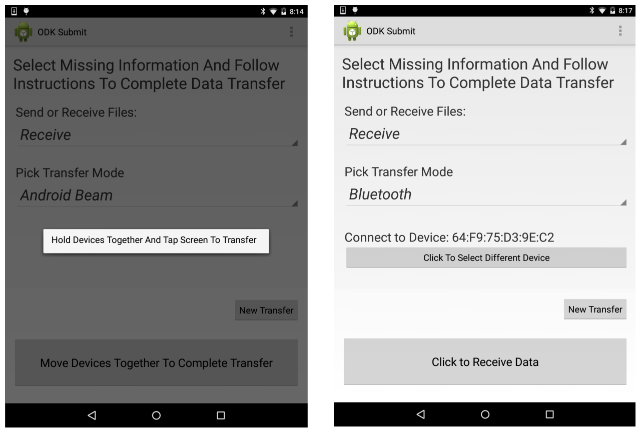
## 4. ODK SUBMIT

To address challenges faced by deployment architects trying to customize their deployments in challenged network contexts, we introduce ODK Submit. Submit combines information from the three perspectives to optimize data transmission in challenged network environments by utilizing multiple heterogeneous networks (e.g., cellular, Wi-Fi, peer-to-peer) based on contextual data properties and available connectivity. Urgent data can be transmitted over more pervasive, higher cost networks, while less urgent data can be transmitted opportunistically over less pervasive, lower-cost networks. ODK deployments are common in rural environments with sometimes sparse connectivity. In a May/June 2015 online survey of the ODK community by Cobb and Sudar et al. [13], 48 out of 56 respondents who deploy mobile devices for data collection reported deployments in rural environments. Respondents from all environments reported that the total size of data collected varied considerably with 19.6% collected gigabytes, 53.6% collected megabytes, 10.7% collected kilobytes of data, and 16.1% did not know. Transmitting gigabytes over expensive connections could potentially be cost prohibitive so understanding how important a piece of data is crucial when deciding to transmit over expensive connections.

The data transmission method selected should be based on an organization's usage model and priorities, as a one size fits all solution will not work for the variety of ICTD use cases. Examples outlined in Section 2.2 have varying requirements from the deployment perspective. For example, forest workers do not have reliable access to electricity, potentially for weeks, making battery life the highest priority. Occasionally replicating data between devices in the field can reduce the chance of data loss. Data importance and other data properties set by a deployment architect should determine when and how much data to replicate to conserve battery life. In contrast, CHWs have better access to power and experience variation in the frequency in which they come into range of inexpensive connectivity (e.g., Wi-Fi at a clinic) and how often they meet other CHWs in the field. As CHWs meet in remote areas, peer-to-peer technologies could be used to transfer data so that a CHW with more clinic visits could act as a Data MULE [25] and transport information with low priority, reducing the amount of data that needs to be sent over costly cellular networks.

To facilitate diverse application requirements, Submit leverages ODK 2.0's XLSX format to specify parameters. ODK users already utilize the XLSX format to specify names of input data fields, define the field's data type, provide question text, flow logic, and other information. We are exploring the use of a basic set of parameters that deployment architects can input into the XLSX system in relative terms (1-10 scale), including data priority (how quickly the data should be transmitted) and data importance (how important it is that the data not get lost). Additionally, we are investigating the provision of deadlines in terms of delta after the data was collected. To minimize the burden on the deployment architect, all parameters are kept optional and - if unspecified - automatically set to lowest values, causing the system to optimize for low-cost routing. Since network costs vary across geographies, Submit requires an organization to create a cost model by specifying a configuration file with cost per byte or cost per message for each transmission medium.

Submit helps to simplify mobile app development by managing connections, providing a user interface (UI), and abstracting various transmission protocol libraries. For long periods in disconnected environments Submit supports peer-to-peer data transfer using Wi-Fi Direct, Bluetooth, both versions of NFC available on Android, and QR Codes. Submit simplifies the process of leveraging Android's peer-to-peer networking capabilities by notifying users when to exchange data and includes abstractions that simplify


ODK Submit
Select Missing Information And Follow Instructions To Complete Data Transfer
Send or Receive Files:
Receive
Pick Transfer Mode
Android Beam
Hold Devices Together And Tap Screen To Transfer
New Transfer
Move Devices Together To Complete Transfer
ODK Submit
Select Missing Information And Follow Instructions To Complete Data Transfer
Send or Receive Files:
Receive
Pick Transfer Mode
Bluetooth
Connect to Device: 64:F9:75:D3:9E:C2
Click To Select Different Device
New Transfer
Click to Receive Data
**Figure 5: Submit's peer-to-peer transmission screen showing an example of NFC (left) and Bluetooth (right) transmission.**

peer-to-peer connection setup. Deployment architects can predetermine parameters for peer-to-peer communication, such as which transfer mode to use. If necessary, Submit presents an unified peer-to-peer interface (shown in Figure 5) to the user prompting them for missing information (e.g., select Bluetooth device to transmit). Additionally, since many synchronization protocols are complex and possibly proprietary, it would be difficult to create a generic tool that conformed to all possible synchronization protocols. Instead, Submit functions as a module that enables developers to flexibly integrate their app-specific protocols with Submit's networking logic. For example, Submit only requires an app to provide contextual properties and metadata (e.g., type, size, priority) and allows a mobile app to maintain ownership of transmission protocols. This means the choice to use Submit does not preclude the use of another sync protocol or encryption schemes as an app can leverage Submit's communication management for only a subset of its data.

### 4.1 System Design

Submit is an Android service that coordinates data communication by providing channel monitoring and transmission scheduling mechanisms to Android apps. For the purposes of this section, the term client app refers to any Android app that binds to Submit's Android service. Submit provides software developers with an interface that abstracts communication channels and flexibly handles data ownership and application-specific synchronization issues. Submit is designed to separate application logic from the network routing logic with a communication system that provides extensibility in terms of: 1) adding transmission channels; 2) modifying transmission channel selection; and 3) handling complex data ownership and application-specific synchronization issues. Submit's service API exposes communication scheduling mechanisms that client apps use to either 1) delegate responsibility of transmitting data to Submit; or 2) register to receive notifications when appropriate network channels are available. When using Submit for notification purposes, a client app takes responsibility to transmit its data with its own possibly proprietary or complex protocol.

Client apps specify two types of objects to interface with Submit: 1) the DataPropertiesObject and 2) the SendObject. The DataPropertiesObject contains metadata that describe the data to be transmitted. The properties supplied include: data size, data urgency, data fragmentability, and reliability requirements. The inherent properties are derived from the perspective of the developer representing "normal" sized data for the client app as an app that primarily transmits responses to survey questions has larger "normal" data than an app that primarily transmits simple reminders the size of SMS

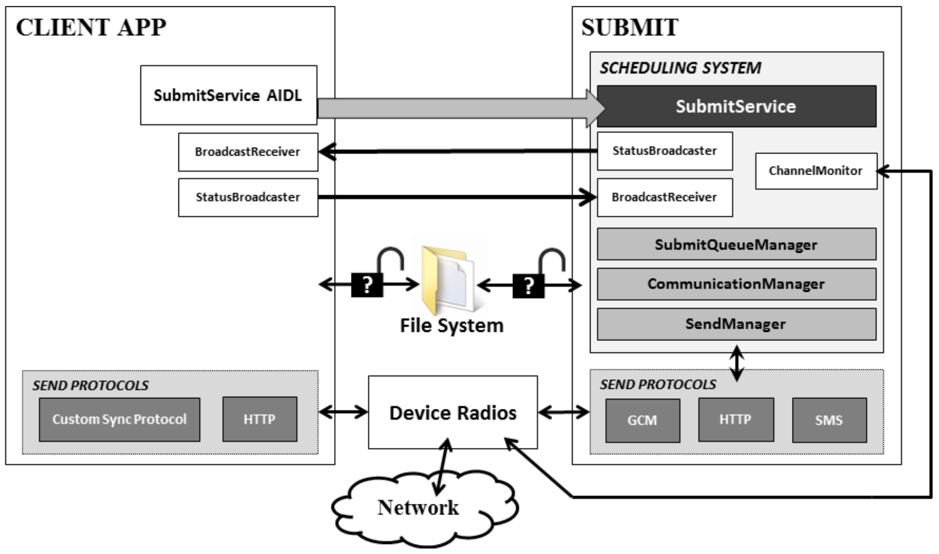


**Figure 6: Architecture diagram showing how Submit interacts with a client app and Android system resources**

messages. By obtaining the software apps perspective on data size, Submit is able to calibrate its routing mechanisms to best handle the common communication case on a per app basis and select appropriate channels. The SendObject contains a list of DestinationAddresses that define the type of transport as well as the necessary parameters to use for the transport. For example, to utilize HTTP POST, the DestinationAddress would contain a URL; to utilize an SMS channel the DestinationAddress would include a phone number. By implementing DestinationAddress as an abstract type, Submit is extensible to various communication protocols. SendObjects also contain the file path to, or string representation of, the data to be sent on behalf of the client app.

Shared data between Submit and client apps could create race conditions as apps are often dependent on their internal data stores being correct and consistent. Submit addresses ownership issues by removing ambiguity through the assumption that the client app owns the data until it explicitly grants Submit temporary ownership rights when it delegates transmission responsibility to Submit. If an app only provides a DataPropertiesObject which has no pointers to the data, Submit assumes the app is maintaining ownership of the data as the client app is only asking for a notification of when to send the data. In contrast, if the client app provides a SendObject containing the actual data or a pointer to an accessible external file, Submit retains ownership of delegated external data until it notifies the client app with the final status of the transmission. Since both the client app and Submit can be responsible for sending data, they must communicate the success or failure of data transmission. Broadcast intents are used to synchronize the sending status. When Submit is responsible for transmitting the data, it broadcasts the status of the communication exchange to the client app. Likewise, if a client app has been notified it is the appropriate time to send the scheduled data, the client app broadcasts the transmission result status to update Submit's internal state.

If the client app delegates responsibility for sending to Submit, the SendManager selects an appropriate network to transmit the data based on the DestinationAddresses and the protocols suited to the available network. By providing multiple client libraries that implement various protocols, Submit increases a client app's ability to communicate using various protocols without requiring expansion of the client app's code base. Currently implemented communication protocols include HTTP/SSL and SMS.

Submit's CommunicationManager is responsible for determining whether an available channel is appropriate for submitted data. The CommunicationManager gauges an available channel's bandwidth capacity and costs. The ChannelMonitor listens to system broadcasts for changes in connectivity, including WiFi events, ad hoc communication opportunities, and cellular events. It reports back the current state of connectivity to the SubmitService when a change is detected. The SubmitQueueManager iterates continually over the pending data that needs to be transmitted (described by DataPropertyObjects). With each pass through the queue, it updates the state of each pending data item based on the results from Submit's protocol modules.

## 4.2 Related Work to Submit

Previous research has explored leveraging a variety of networks for data transmission [8] and splitting data over multiple networks based on cost and availability [9, 18, 21]. While work exploring simultaneous data transmission over multiple interfaces has been shown to improve mobility, power efficiency, and network capacity [8], our work focuses on selecting a single network from a heterogeneous combination of different transmission opportunities in a manner specialized to the deployment context. Submit is most similar to work that focuses on identifying the best type of network for data transmission given various contexts and policies. MultiNets [23] proposes real-time switching between different network interfaces on mobile phones using policies based on power, data offloading, throughput, and latency. However, policies are configured by the user and are applied to every app that uses the device. In contrast, Submit provides a library that allows deployment architect to configure policies that will only be applied to apps relevant to the application. In this way, Submit is most similar to Delphi [15], a transport layer module that selects the most appropriate network for data transmission given policies set by applications. However, Delphi assumes operation in an environment with ubiquitous connectivity and focuses on the transport layer not the application layer. While it attempts to provide a systematic evaluation of various networks for data transmission, it does not address many of the issues of developing contexts including intermittent connectivity and regional pricing policies. Also in contrast to Delphi, Submit also uses information about data (e.g., time-sensitivity, importance) to identify the best method for transfer. Haggle [24] is another solution that separates application logic from inflexible pre-programmed transport bindings so that applications can communicative in dynamic networking environments. Haggle and Submit both provide API's to developers but Submit goes further and

Table 2: Average latency for a client app sending data using Submit and without using Submit

| | Wi-Fi w/ Submit | Wi-Fi w/o Submit | 3G w/ Submit | 3G w/o Submit |
|---|---|---|---|---|
| 10 KB | 0.12s | 0.10s | 0.59s | 0.47s |
| 100 KB | 0.40s | 0.28s | 1.66s | 1.28s |
| 1 MB | 2.53s | 2.03s | 10.93s | 7.86s |
| 10 MB | 22.55s | 20.58s | 83.95s | 80.35s |

provides constructs for deployment architects who are not programmers to adjust their composable mobile information system. Another issue with Haggle is that it proposes a general form of a naming notation to allow for late-binding that is independent of the lower-level address. While a good idea, the infrastructure is not currently available to support some of these assumptions. In contrast, Submit is designed to work with existing infrastructure to enable applications to "just work" in different environments with limited infrastructure. Our work distinguishes itself from previous research in that it seeks to provide a network management module at the application layer that enables flexible control to an organization deploying a customizable app in areas of limited connectivity.

## 5. EXPERIMENTS

### 5.1 Splitting Data Transmission

To measure the baseline impact Submit has on a client app's communication performance, we integrated a simple file upload app with Submit to evaluate network usage and latency. The test involved the client app using HTTP POST to upload data from a client to a server with and without Submit. Performance was measured in two scenarios: Wi-Fi only and 3G only. The tests were performed on a Samsung Galaxy running Android 4.3 using either 3G or Wi-Fi networks. The results in Table 2 show the average latency for each file size for the ten uploads. As expected, there is slight latency overhead associated with Submit due its use of remote procedure calls and broadcast intents to communicate with a client app for each uploaded file.

Submit's latency additions are counterbalanced with its ability to minimize network usage according to user preferences. To verify the reduction of cost for network usage a small experiment was performed for ten minutes where Wi-Fi was disabled, leaving only 3G accessible. In this experiment, the client app uploaded a randomly selected file between 5 KB and 100 KB from a directory of files. For the client using Submit, a threshold value was set that prevented any file over 7 KB from being sent over a mobile broadband network. After ten minutes, the Wi-Fi was re-enabled for 10 minutes. After the entire 20 minutes the number of packets sent over Wi-Fi was compared to the more costly 3G network. The client without Submit sent over 380.6 KB over 3G whereas the client using Submit only sent 12.8 KB over 3G. Thus, Submit selectively uses one network while waiting for cheaper channel to become available.

To understand Submit's effects on deployment scenarios a small sampling of 85 actual site visit records were used to calculate data transmission reductions for the Site Visit scenario described in section 2.2. Table 3 shows the calculated average reduction if Submit managed ODK Collect's data submission process. By simply separating transmission of the text portion of the data and delaying the transmission of the photo's binary data until the device is in free Wi-Fi range would mean 0.37% of the total data would be transmitted via cellular and 99.53% of the data would be transmitted over Wi-Fi. Using Submit's concept of data priority could further reduce cellar transmission to 0.1% of total bytes by only transmitting the information about medication inventory.

Table 3: Reduction of transmission if record is split by data type or data priority for site visits scenario

| | Bytes | Percent |
|---|---|---|
| Avg Total Record Transmission Size | 330,773 | 100.00 |
| Avg Data Size | 1,213 | 0.37 |
| Avg Photo Size | 329,217 | 99.53 |
| Avg Priority Data Size | 343 | 0.10 |

### 5.2 Peer-To-Peer Transmission

We evaluate the performance of peer-to-peer transmission methods by comparing 5 methods of transferring data between Nexus 7 devices positioned 0.5 meters apart running Android 4.4.4. WiFi Direct and Bluetooth were tested using transfer sizes of 1 KB, 10 KB, 100 KB, 1 MB, 10 MB, 100 MB, and 1 GB of data. NFC transfer sizes were limited to data transfer times that were feasible for generic peer-to peer use (large transfers took too long). NFC with Bluetooth was tested up to 10 MB, while NFC only was tested up to 100 KB. Additionally, peer-to-peer transfer using QR codes was tested by having one Nexus 7 display QR Codes on its screen while another Nexus 7 read the QR codes with its built-in camera from 0.3 meters away. The ZXing library was used to generate and scan the QR Codes. While QR Codes specifications state transmission up to 4 KB of data [2], our results show that transfers are unreliable past 1KB of data. The time it takes to scan a QR Code is fairly consistent as the size of data increases, but the error rate increased to over 60% for file sizes larger than 1 KB.

Bluetooth and NFC were the fastest transfer options for smaller amounts of data as shown in Figure 7. As the data size increases WiFi Direct emerges as the fastest mode of transfer. Until data sizes exceed 100 MB, the total time for WiFi Direct remains essentially constant because establishing the connection dominates the transfer time [12] as shown in Figure 8. WiFi Direct is a realistic choice for data on the order of 1 MB or larger, for anything lower than 1 MB Bluetooth may be a better option. Figure 7 also shows that NFC only is significantly slower than Bluetooth enabled NFC[4]. QR scanning had the largest variance in the duration of file transfer. While increasing the error correction level of the code can help remedy this issue, for data sizes close to 1 KB the QR Code was too dense to accurately and consistently be scanned.

Since battery life is important in disconnected environments, we evaluated the power performance of WiFi Direct, Bluetooth, and QR codes. For each test a connection was established between two fully charged Nexus 7's and data was continuously transmitted from one device to the other until the sending device's battery was depleted. For consistency the device screens remained on at all times since the QR method requires the screen to be on and active usage of devices would cause the screen to be active some percentage of the time. The experiments revealed that despite being able to leave the device in airplane mode, the QR Code scanner consumed more battery than traditional data transfer methods. The QR scanner took 6.8 hours to drain the battery to a 10% level while WiFi direct transfer only lasted 0.33 hours longer. In comparison it took about 9.3 hours of Bluetooth transmission to drain the battery to a 10% level. The results suggest that the power required to continually use the camera and process bar codes resulted in greater battery consumption over time than WiFi or Bluetooth transmission.

The main factors for selecting a peer-to-peer method include transfer time and battery efficiency. Per byte, WiFi is more battery efficient, but within the range of 100 KB to 1 MB, Bluetooth is faster. In the case of forest inventory workers, opportunity to charge the devices is the limiting factor and WiFi should probably be selected, since it is more efficient per byte. For the CHWs, avoiding the slightly more cumbersome connection process of WiFi might

[4]http://developer.android.com/training/beam-files/

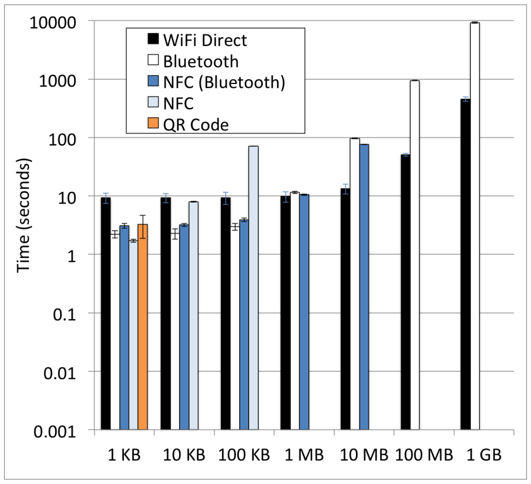


**Figure 7: Data transfer times associated with peer-to-peer technologies with different file sizes. (Log Scale)**

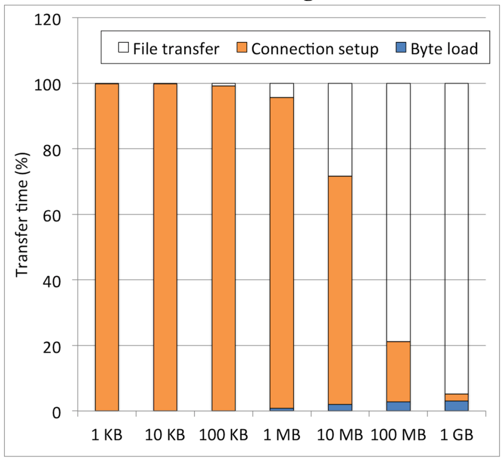


**Figure 8: Percent of time spent in different phases of WiFi Direct transfer. Connection setup time dominates small file size transfer.**

be more important. The main disadvantage of both Bluetooth and WiFi Direct is the difficulty for the user to confirm which device they connected to. While this may be less of an issue in a forest inventory setting, it is one of the primary concerns in a clinical setting. With both Bluetooth and WiFi Direct, someone attempting to steal data can spoof their device name and MAC address, potentially deceiving a user. NFC and QR Code communication allows users to visually clarify that the correct device is receiving the data. This can be an important advantage when the information is confidential such as medical data. The results show that the QR Code scanner is slower and less power efficient than NFC with Bluetooth. However, there is the possibility of hand-to-hand contact from using NFC when the two devices are brought close together to establish the connection. Hand contact could be a disadvantage in a remote clinical setting where hygiene practices might restrict such contact. A key advantage of Bluetooth over WiFi Direct is the ability to pair devices ahead of time, allowing users to more confidently send their data to the correct person. However, if NFC is not an acceptable option due to hand-to-hand contact or data size, white-listing Bluetooth devices could increase security in a clinical setting.

## 5.3 Usability of Peer-To-Peer Transfer

To understand the overhead of using different peer-to-peer modalities we conducted basic usability tests with 22 participants. The participants' ages ranged from 18 to 56, with a mean age of 25. After initial demographic information was collected, participants were given a short training session on how to use Submit's manual peer-to-peer transfer screen. Participants were then given a list of ten 1KB transfers tasks to complete, one sending and one receiving for each of the five transfer methods. The order of the task list was randomized across participants so that each transfer method appeared with similar frequency at each position. The first two tasks for each user used a specific transfer mode (e.g., send using NFC, then receive using NFC). After the first two tasks, participants were asked to complete the NASA TLX[5] form to rate the difficulty of the specific transfer mode. Once completed, participants proceeded with the eight remaining tasks. After the ten tasks were completed, a semi-structured interview was used to solicit feedback about the most confusing part of the transfer process and to help identify possible improvements to Submit's peer-to-peer transfer. Participants were also asked to rank the five transfer methods based on ease of use (on a rank scale from 1 to 5, where 1 was the easiest transfer method to use and 5 was the most difficult), and rank them based on efficiency (on a rank scale from 1 to 5, where 1 was quickest transfer method and 5 was the slowest method).

Usability results confirmed the results from transmission performance. Users found that using a QR Code to transfer data was both the least efficient ($p < 0.001$) and the most difficult method ($p < 0.001$) to use[6] with a mean efficiency rank of 4.7 and a mean difficulty rank of 4.6. In three cases, users were unable to successfully scan the QR Code due to the lighting conditions of the room. Users also found that Bluetooth and WiFi Direct were the fastest ($p < 0.001$) and easiest ($p < 0.001$) to use.[6] WiFi Direct had a mean efficiency rank of 2.3 and a mean difficulty rank of 2.3. Bluetooth had a mean efficiency rank of 1.9 and a mean difficulty rank of 1.8.This slightly differs from the data gathered during channel testing. Due to the longer connection setup time, WiFi Direct should have been outperformed by all four of the other transfer methods. This discrepancy can be explained by the fact that users had to select more information during the user testing. The time it takes to select all this information causes the actual speed of transfer to matter less when compared to the overall time spent using the application. Additionally, the time it takes to move the devices together for NFC, and the time it takes to line up the devices for QR Code scanning was not accounted for in the evaluation of performance. These usability issues dramatically increase the overall time it takes to transfer using NFC and QR Codes. Testing results showed QR Codes take an unreliable amount of time due to user and environment conditions such as reflections due to lighting. However, based on the data collected from the NASA TLX form, there was no significant difference between the average level or type of stress users experienced while using the different transfer modes ($p = 0.57$)[7]. This implies that while users do prefer some transfer methods above others, the difference between them is relatively small compared to the overall ease.

While users found Bluetooth and WiFi Direct to be the fastest and easiest methods and QR Codes to be the slowest and most cumbersome, results were more varied for the NFC options. Some users liked the strong visual and physical cues associated with holding the devices together and touching the screen to beam the data between devices. Users also appreciated that the receiving NFC user does not have to select any options since the parameters, such as from whom you are receiving, are all automatically inferred

[5]http://humansystems.arc.nasa.gov/groups/tlx/
[6]Significance calculated using the Kolmogorov-Smirnov Test.
[7]Significance calculated using the Kruskal-Wallis Test.

when you hold the devices together. Other users felt uncomfortable with the inevitability of hand to hand contact that comes from holding the devices together with NFC. Other users expressed concern surrounding potentially dropping the device. Most users held the tablets together with one hand, and tapped the screen with their other hand. They thought NFC was slightly more inconvenient than other mediums, but would not mind using it.

## 6. CONCLUSION

Configurable mobile data transmission frameworks can enable deployment architects with limited programming skills to adapt mobile devices to meet diverse application requirements in challenged network environments. High-level data and networking abstractions can improve mobile app design paradigms by making single-purpose apps more malleable to resource-constrained contexts where issues such as affordability, infrastructural constraints, institutional capacity, and technical support are nontrivial. Submit improves deployments by supporting variations in deployment contexts in a systematic manner. Submit's abstractions decouple data and connectivity to enable application-level optimization of sparse heterogeneous networks. Submit identifies available connectivity in challenged networking environments and sends appropriate data over available channels in limited resources settings.

Experiences from the field highlight the fact that the selection of appropriate technologies for data transmission involves accounting for inherent data properties, contextual data properties, and network properties. Submit enables deployment architects to shape the communication priorities of the components comprising a larger application. Several deployment scenarios of interest benefit from peer-to-peer connectivity when centralized infrastructure is unavailable for data transmission. Which peer-to-peer transfer method to use should also be based on inherent properties (e.g. data size), contextual properties (e.g. data importance, security), and battery constraints. Our experiments showed the most significant barrier to peer-to-peer transmission is the time it takes for users to set up a peer-to-peer connection. In order to address these challenges, Submit handles peer-to-peer connections internally and automatically populates most settings before providing a unified UI to the user.

Creating software tools that enable application-level communication optimizations through the selection of appropriate data for transmission over available network channels represents a necessary complement to infrastructural improvement. Data communication needs to be adaptable to deployment conditions and solutions that focus on optimizations to the network transport layer do not have the flexibility to leverage the sparse challenged network conditions that exist. Therefore, adaptable frameworks that create abstractions that target application-level users (as opposed to developers adapting to network transport layers) are needed to empower deployment architects to easily customize application deployments to match an organizations requirements. For mobile tools to be successful in resource-constrained environments they should be composable by non programmers, deployable by resource-constrained organizations, usable by minimally trained users, and robust to intermittent power and networking outages.

## 7. ACKNOWLEDGMENTS

We thank Samuel Sudar for his help on this project. We also thank Neha Kumar and Elizabeth Belding for their insightful feedback on the paper. The material in this paper is based upon work supported by NSF research grant IIS-1111433, NSF Graduate Research Fellowship grants DGE-0718124 and DGE-1144085, and USAID contract AID-OAA-A-13-00002.

## 8. REFERENCES

[1] Obami. http://www.obami.com/portals/obami/about_obami.

[2] Qr Code. http://www.qrcode.com/en/about/version.html.

[3] The World in 2011: ICT Facts and Figures. https://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2011-e.pdf, 2011.

[4] My MedLab. http://www.mymedlab.com/, 2013.

[5] Tribal Initiatives. http://transition.fcc.gov/indians, Mar 2013.

[6] The World in 2013: ICT Facts and Figures. https://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2013-e.pdf, 2013.

[7] M-pesa. http://www.safaricom.co.ke/m-pesa/, 2014.

[8] P. Bahl, A. Adya, J. Padhye, and A. Walman. Reconsidering Wireless Systems with Multiple Radios. *SIGCOMM Computing Communication Review*, 34(5):39–46, October 2004.

[9] A. Balasubramanian, R. Mahajan, and A. Venkataramani. Augmenting Mobile 3G Using WiFi. In *Proc of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 209–222, 2010.

[10] E. Brewer et al. The Challenges of Technology Research for Developing Regions. *IEEE Pervasive Computing*, 5(2):15–23, 2006.

[11] W. Brunette et al. Open Data Kit 2.0: Expanding and Refining Information Services for Developing Regions. In *Proc of the 14th Workshop on Mobile Computing Systems & Applications*, HotMobile '13, 2013.

[12] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano. Device-to-Device Communications with Wi-Fi Direct: Overview and Experimentation. *Wireless Communications, IEEE*, 20(3):96–104, June 2013.

[13] C. Cobb, S. Sudar, R. Anderson, F. Roesner, and T. Kohno. Work in progress, 2015.

[14] L. B. Deek, K. C. Almeroth, M. P. Wittie, and K. A. Harras. Exploiting Parallel Networks Using Dynamic Channel Scheduling. In *Proc of the 4th Annual International Conference on Wireless Internet*, WICON '08, pages 1–9, ICST, Brussels, Belgium, Belgium, 2008.

[15] S. Deng, A. Sivaraman, and H. Balakrishnan. All Your Network Are Belong to Us: A Transport Framework for Mobile Network Selection. In *Proc of the 15th Workshop on Mobile Computing Systems & Applications*, HotMobile '14, 2014.

[16] B. DeRenzi et al. E-IMCI: Improving Pediatric Health Care in Low-income Countries. In *Proc of the Conference on Human Factors in Computing Systems*, CHI '08, pages 753–762, 2008.

[17] B. DeRenzi et al. A framework for case-based community health information systems. In *Global Humanitarian Technology Conference (GHTC), 2011 IEEE*, pages 377–382. IEEE, 2011.

[18] B. Han et al. Mobile Data Offloading Through Opportunistic Communications and Social Participation. *IEEE Transactions on Mobile Computing*, 11(5):821–834, May 2012.

[19] C. Hartung et al. Open Data Kit: Tools to Build Information Services for Developing Regions. In *Proc of the 4th ACM/IEEE Int Conf on Information and Communication Technologies and Development*, ICTD '10, 2010.

[20] R. Heeks. Avoiding eGov Failure: Design-Reality Gap Techniques. www.egov4dev.org/success/techniques/drg.shtml, 2008.

[21] Y. Li, G. Su, P. Hui, D. Jin, L. Su, and L. Zeng. Multiple Mobile Data Offloading Through Delay Tolerant Networks. In *Proc of the 6th ACM Workshop on Challenged Networks*, CHANTS '11, pages 43–48, 2011.

[22] M. Martins and R. Fonseca. Application Modes: A Narrow Interface for End-user Power Management in Mobile Devices. In *Proc of the 14th Workshop on Mobile Computing Systems and Applications*, HotMobile '13, 2013.

[23] S. Nirjon et al. MultiNets: Policy Oriented Real-Time Switching of Wireless Interfaces on Mobile Devices. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2012 IEEE 18th*, pages 251–260, April 2012.

[24] J. Scott et al. Haggle: A Networking Architecture Designed Around Mobile Users. In *WONS 2006: Third Annual Conference on Wireless On-demand Network Systems and Services*, pages 78–86, 2006.

[25] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: Modeling and Analysis of a Three-tier Architecture for Sparse Sensor Networks. *Ad Hoc Networks*, 1(2âĂŞ3):215 – 233, 2003.

[26] K. Shirima et al. The Use of Personal Digital Assistants for Data Entry at the Point of Collection in a Large Household Survey in Southern Tanzania. *Emerging themes in epidemiology*, 4(1):5, 2007.

[27] R. W. Stevens and G. R. Wright. TCP/IP Illustrated: Vol. 2: The Implementation, 1995.

[28] M. Vitos, J. Lewis, M. Stevens, and M. Haklay. Making Local Knowledge Matter: Supporting Non-literate People to Monitor Poaching in Congo. In *Proc of the 3rd ACM Symp on Computing for Development*, ACM DEV '13, 2013.

[29] L. Wei-Chih et al. UjU: SMS-based Applications Made Easy. In *Proc of the First ACM Symposium on Computing for Development*, ACM DEV '10, 2010.

[30] K.-K. Yap et al. Making Use of All the Networks Around Us: A Case Study in Android. In *Proc of the 2012 ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design*, CellNet '12, pages 19–24, 2012.