Distributing Private Data in Challenged Network Environments

Azarias Reda
University of Michigan
Ann Arbor, Michigan
azarias@umich.edu

Brian Noble
University of Michigan
Ann Arbor, Michigan
bnoble@umich.edu

Yidnekachew Haile
HilCoE College
Addis Ababa, Ethiopia
yidnekr@yahoo.com

ABSTRACT

Developing countries face significant challenges in network access, making even simple network tasks unpleasant. Many standard techniques—caching and predictive prefetching—help somewhat, but provide little or no assistance for *personal data* that is needed only by a single user. Sulula addresses this problem by leveraging the near-ubiquity of cellular phones able to send and receive simple SMS messages. Rather than visit a kiosk and fetch data on demand—a tiresome process at best—users request a *future visit.* If capacity exists, the kiosk can schedule secure retrieval of that user's data, saving time and more efficiently utilizing the kiosk's limited connectivity. When the user arrives at a provisioned kiosk, she need only obtain the session key on-demand, and thereafter has instant access. In addition, Sulula allows users to schedule data uploads. Experimental results show significant gains for the end user, saving tens of minutes of time for a typical email/news reading session. We also describe a small, ongoing deployment in-country for proof-of-concept, lessons learned from that experience, and provide a discussion on pricing and marketplace issues that remain to be addressed to make the system viable for developing-world access.

Categories and Subject Descriptors

C.2.5 [**Computer Communication Networks**]: Local and Wide-Area Networks—*Access schemes*; D.2.8 [**Software Engineering**]: Software Architectures; C.2.4 [**Computer Communication Networks**]: Distributed Systems—*Client/server*

General Terms

Design, Performance, Economics

Keywords

Personal Data, Developing Regions, WWW access, Bandwidth, Latency, SMS, Limited Connectivity, Caching, Prefetching, Ethiopia

1. INTRODUCTION

One of the major indicators of the digital divide in developing countries is the serious lack of network resources. Connectivity is very expensive, and there is an order of magnitude difference between the end user bandwidth available in developing countries and the developed world. A 1 Mbps connection in the US costs about 25 USD per month, whereas a similar connection in Zambia costs about 1,700 USD per month [24]. While there are many reasons for this, such as expensive international bandwidth, inefficient government monopolies, lack of proper peering points, low user density and poor provisioning, the bottom line is users have an unpleasant experience when using the internet for anything that involves more than a few hundred kilobytes of data transfer [5]. During our recent visit to Ethiopia, checking our university email and sending a few pictures back home was usually an affair that took most of an afternoon. Even such slow access is not ubiquitous. Connectivity is often provided through internet kiosks or public libraries that usually have long waits, with the situation exacerbated by the slow, shared dialup connections that have effective bandwidth delivered to a user hovering around 10-15 Kbps [30].

There have been several attempts to address this problem, ranging from opportunistically prefetching content and storing it for later use [14] to employing peer-to-peer technology to cooperatively download data files and cache them for other users [33]. However, these solutions target data that is requested and consumed by multiple users. Each leverages redundancy in data requests to avoid fetching a piece of data several times, and delivers it from a local cache whenever somebody else requests it, thereby making future requests fast. This is a tried and true approach, used in well connected environments by content distribution networks and most modern browsers. However, this solution cannot help with personal and private data—unfortunately, such data comprises a substantial portion of that consumed in internet kiosks [8, 31, 38]. In other words, when the data is either personal or only personally interesting, such as email or a school application, a user can not expect other people to have prefetched the data for her. In these cases, users must wait while their data is being fetched from a remote host over a very slow link.

We propose a solution to this problem by extending the idea of individualized content distribution networks [6] that take personal usage patterns into account. Our system, called *Sulula*, provides an infrastructure for distributing private data in challenged networks. We leverage the wide penetration of cellular telephones in developing countries to securely deliver private data to nodes in low bandwidth, high latency environments ahead of usage time. By the end of 2008, there were more than 4 billion mobile telephone subscribers around the world, with Africa having the highest mobile growth rate [18]. In Ethiopia, for example, there are about 25X as many mobile subscribers compared to internet subscribers [11]. This allows us to take advantage of widely available services like SMS in our design. To motivate our approach and better explain our solution, consider the following usage scenario.

Our user, Elsa, is a bank teller in Addis Ababa, Ethiopia. Elsa spends most of her day at work, and she also takes classes a few nights during the week. She does not have a car and uses public transportation to get to work and back. She has some time during the day when she takes breaks from work, and some more time in the evenings. However, the minibuses she takes home stop running once it gets darker.

Elsa knows of three internet kiosks that she can reasonably get to. One is within walking distance of her work(K_A), one is on her route to work(K_B), and one is close to her home(K_C). It is impractical to check her email and do other work at K_A because the 15-30 minutes break she takes during the day is never enough to walk over there and even properly load her emails. She could break her trip home into two, and stop by K_B, but by the time she is done, it might get too dark, and she might miss the minibuses going home. The only viable alternative is K_C because it is closer to home and she can go there after work. However, that is also true for many others and there is a long wait there. Even after that, there are many people using the lines and the connections are terribly slow.

Elsa, like most people in the cities, has a basic cell phone to make calls and send SMS messages. This is what our solution takes advantage of. In our system, she keeps the phone numbers of the three kiosks in her phonebook. About an hour before her break during work, she sends an SMS to the kiosk system at K_A, with her estimated time of arrival, thereby requesting her data to be fetched for her. After looking at the allocated resources, the system informs her whether her request can be serviced or not. If not, it also suggests the earliest available time at which it could be. At the scheduled time, she walks over to K_A and provides her password to access her data, which is ready and waiting for her. She could also do the same with any of the other kiosks depending on her daily plans and routes. By providing an efficient and secure way to deliver private data in challenged networks, our system eliminates most of the wait time when using the internet in those environments. Using service plugins, or *channels*, our infrastructure is able to deliver data from various sources such as email providers, RSS feeds, web portals etc. It also allows internet kiosks to better manage the scarce network resources and provide more value to their customers.

The main contribution of this work is a framework for distributing private data securely to nodes in challenged network environments that leverages usage patterns and an advance notification mechanism. In addition, we implement three different channels that use our framework to deliver personal and personally interesting data from various services. At the time of this writing, we also have a test deployment running in Addis Ababa, Ethiopia.

2. DESIGN

Sulula provides a framework for distributing personal data in challenged networks environments. At the core of the system is a data transfer infrastructure between two end points with the following properties:

- Supports an advance alert mechanism, triggering data transfer at the request of the owner.
- Supports resource management with a scheduling component that factors in available network resources and priority.
- Operates in challenged networks with low bandwidth and high latency, delivering data ahead of expected need.
- Provides a content agnostic interface allowing applications to embed their own semantics in their data.
- Provides confidentiality, integrity and source authentication - necessary properties for secure transmission of private data.

All data in the system is associated with an owner, which is represented by an opaque ID that is consistent across the system. A typical usage pattern for our system involves data creation or aggregation at the source, data request by the user, data delivery through the Sulula infrastructure and finally data consumption. We will first discuss some of the guiding principles we considered in designing the system.

In the expected usage pattern, users know ahead of time where they want their data delivered, and provide a hint to the system to initiate the data transfer. However, a user is not always guaranteed to consume the data requested, and as a result, there might be cases where a delivered piece of data is never consumed or consumed in time. In addition, most of the data that flows through our system has a certain freshness constraint, which makes data interesting only within a given time frame. Having a time-to-live associated with all delivered data enforces these constraints. In addition, it relieves the data source from having to keep state information about old data, which in most cases is not valuable to the user in any event.

Freshness of data in our case is defined relatively. The nature of challenged networks makes it infeasible to transfer real time, or near real time data. Sulula is best suited for data that stays fresh longer than it takes to deliver it. Our system leverages this to schedule transfers based on available resources and expected consumption time of the data. By providing metadata about the data to be delivered, the source assists the resource management and scheduling at the data consumption point. When requested data is ready to be delivered, the data source packages it by compressing and encrypting it. This package is delivered to the consumption point along with the time-to-live. By transferring the bulk of the data before a user gets to the consumption point, the interactive operation that requires a network connection once the user arrives is reduced to obtaining a session key for the encrypted content. Keys are distributed to consumption points only upon the request and authentication of users.

So far, we have described the properties of the data source and the consumption point. A consumption point is where the user interacts with our system. In challenged network environments, this could be an internet kiosk or a library

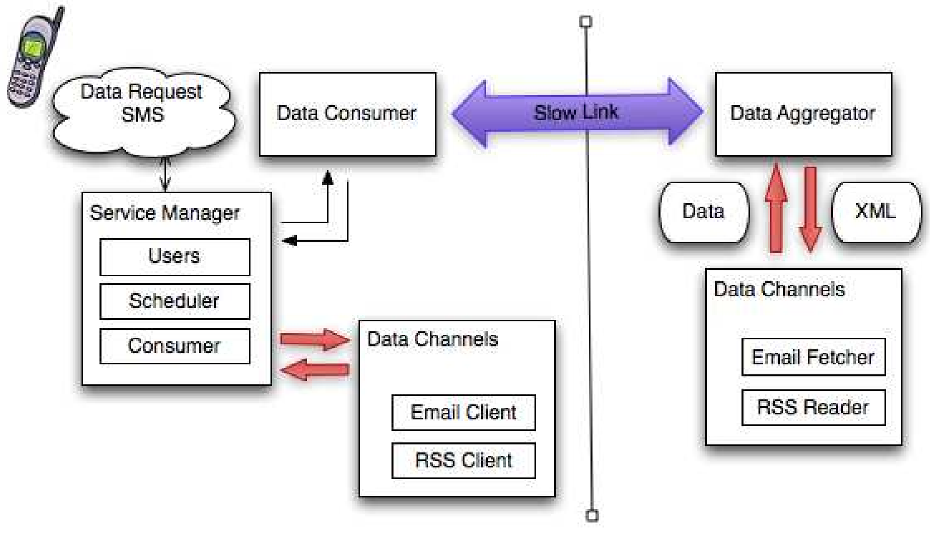

Figure 1: The Sulula Infrastructure

computer that a user has access to. On the other hand, a data source is any service that has some data of interest to a user. This could be an email provider, a social network site, a blog etc. that has either personal or personally interesting data available for the user. Obviously, we can not expect existing services to adopt support for our system. Instead, we rely on an aggregator, which supports our interface, but also knows how to interact with existing services, and puts relevant data in the system to be accessed by clients.

Our system supports data aggregators by way of data channels. A data aggregator is a server located in a well connected environment that supports the Sulula interface and collects data that is of interest to users. Data collection is done using data channels, which are enabled by plugins that run at the aggregator and at the data consumption end. An email channel, for example, could aggregate a user's email at the server and make it available to the data consumption end using the Sulula infrastructure. Figure 1 shows the Sulula system with two implemented channels.

2.1 SMS

The International Telecommunication Union's 2009 ICT development index [18] reports that by the end of 2008, there were over 4 billion mobile cellular subscribers worldwide, translating into a penetration rate of 61%, which compares to just about 1.3 billion fixed telephone lines - or 19 per 100 inhabitants. This makes mobile phones easily the most widespread modern technology the world has ever seen. Furthermore, mobile phones may well have the greatest impact on development [9]. In fact, the ICT index states that over 60% of the world's mobile subscribers were from developing countries, with Africa having the highest mobile growth rate—one in four people have a mobile phone, significantly higher than the Internet penetration.

Using SMS to alleviate the acute lack of network resources, therefore, leverages this wide penetration. As most internet consumers also have access to a mobile phone (close to a 100% in our user study described in section 5), integrating SMS is not a difficult transition. Having an SMS interface for requesting transfers, rather than something like a fixed schedule of data delivery, allows the users to choose when they want to go to internet kiosks without having to deal with stale data, and gives them the option of comparing multiple service providers before committing to buy a service. On the other hand, it is also important to make adoption by kiosks and other internet providers as simple as possible. For this reason, we use a smartphone connected to a commodity PC to serve as an SMS gateway that can receive and send messages while interacting with applications running on the PC. In our deployment, we are using a second-hand smartphone bought for less than $50. Such a low price point reduces the barrier to entry for business owners and encourages them to offer Sulula as an option. In addition, we imagine this phone could also double as the kiosk cell phone as most kiosks have an additional business letting people make long distance calls.

2.2 API

To accomplish the above properties, Sulula provides a pair of API's, one at source of the data, and another at the consumption point, which expose the following functionalities. At the data source end:

- `Add Owner` - Adds an `<owner>`, a unique opaque identifier, to the system. Data is private to its owner.
- `Put Data` - Inserts data for a given user.
- `Ready Data` - Packages (compresses and encrypts) all available data for a user, as well as provides information about the data.
- `Fetch Data` - Delivers data from the source for an owner. This data is encrypted and has a specified time to live, after which point the key becomes invalid.
- `Get Key` - Provides a key for a data transaction by owner, as long as the request is made within the TTL given in the `Fetch Data` request.

At the consumption point:

- `Request Transfer` - Enables an `<owner>` to request for scheduling of data transfer to a particular consumption point. This is exposed through an SMS hook for delivery scheduling, and could also be a network message.
- `Initiate Transfer` - Contacts the data source about available data for an owner and schedules its delivery, and provides information about earliest availability time of data.
- `Ready Data` - Makes data ready for consumption, decrypted and in the state it was stored in `Put Data`.

To enable communication between data channel pairs at the source and consumption point, our system supports two additional operations:

- `Add Channel` - Each `<owner>` can have any number of supported channels associated. Channels are uniquely identified and are described as plugin specific XML messages that enable data collection and aggregation.
- `Deliver Plugin Data` - Plugins can send chunks of data asynchronously to their counterparts on the server. Such data is associated both with the owner of the data and the particular plugin that generated it, and is delivered to the server based on a system wide scheduling of available network resources.

A fully implemented channel provides three operations:

- At the consumption point:
  - Add corresponding channels to the system (with channel details crafted as XML strings that include the information the server component of the plugin would need to collect user data)
  - Given the user data, be able to attach semantic meaning and display it to the user. This often could be as simple as invoking a browser with HTML data, but could also involve presenting email messages, RSS feeds etc.
- At the aggregator end:
  - For a corresponding channel, `Put Data` in the Sulula system for the owner.

To better explain how channels function in the Sulula system, lets look at how an RSS channel might work. Let's say the system has user `<Elsa>`.

- At the users end point, an RSS plug-in would use `Add Channel` to add the following for `<Elsa>`
  - `(RSS, "<RSS feed='www.umich.edu/press.xml'> </RSS>")`
  - `(RSS, "<RSS feed='www.someblog.com/feeds' user='foo' pass='bar'></RSS>")`
- At the data aggeregator, an RSS server-plug-in would use this information to fetch and package instructed data and use `Put Data` to place it in the system.
- Finally, when data is ready for consumption at the user's end, the system uses the channel plug-in to display it to the user (again, often with a browser).

2.3 Uploads

In addition to requesting future visits, Sulula allows users to schedule data uploads. Scheduling such uploads is delegated to data channels. Consider an email channel, for example. When the user is finished with an email session that might have included reading, tagging and composing mail, the channel compiles all data that needs to reach the aggregator so that further network action, such as delivery to the SMTP server, can take place. The channel then uses the `Deliver Plugin Data ()` API call to pass this data to the Sulula infrastructure, which schedules it for asynchronous delivery to the aggregator. The channel counter part on the aggregator can further process this data and communicate the results to the kiosk email client, such as the successful transfer of mail to the recipient.

2.4 Capacity

The ability of consumption points to service user requests is an important factor in determining the practicality of our system. Our week long observation of a medium sized internet kiosk in Addis Ababa with five PCs and two dial up lines, as summarized in table 1 below, shows that kiosks are busier during the late afternoon and early evening hours as students and workers make up most of the consumer base. This leaves ample opportunities for the kiosk to queue and service requests from Sulula users through the rest of the day. In addition, dial up fees are significantly cheaper during off-peak hours (for example, about half off for the 6 pm - 8 am block of the dial-up service provided by the Ethiopian Telecommunication Corporation [10], a monopoly within the country). This increased capacity enables kiosks to take requests for overnight deliveries, at possibly reduced prices. Finally, Sulula supports a transparent scheduling system that notifies users the earliest possible time data would be available if their request can not be serviced by the desired time, which allows users to weigh their options from multiple providers. This could be especially useful when some of the kiosks are already heavily subscribed.

Table 1: Customer Load

Two hour block	# of customers
8am - 10am	2
10am - Noon	2
Noon - 2pm	2
2pm - 4pm	2
4pm - 6pm	3
6pm - 8pm	5
8pm - 10pm	5

3. IMPLEMENTATION

Sulula is implemented as a pair of services that run at the data source and consumption point, and communicate using XML-RPC [41]. Both ends provide API's that plugins and applications can use to deliver data. The data source was implemented in a UNIX environment for seamless integration with most online services while the data consumption end runs on a Windows environment for ease of use at internet kiosks and other service providers. Using the provided plugins, the server can also double as a data aggregator. As we will describe shortly, data channels are implemented as simple modifications to existing applications that are used to collect and consume private data.

The data source service consists of data processing, plugin manager and XML RPC server components. The data processing module is responsible for packaging and encrypting data and making it ready for delivery. We use the 128-bit Advanced Encryption Standard (AES-128) for block encryption and MD5 for hashing. These could easily be replaced with other encryption and hashing mechanisms. The XML RPC server runs on a well-known port and can handle multiple requests simultaneously. The plugin manager interacts with channel plugins, and provides the interface plugins use to manipulate and deliver data to users.

The data consumer service is implemented as a standalone multithreaded process that accepts requests from users using an SMS interface, and interacts with plugins as well as the data source. It was implemented on a Windows platform as Windows represents the overwhelming majority of operating systems run on computers used in developing countries [28]. We built our system using the .NET 2.0 framework, and make use of an open source XML-RPC package [3] to interact with the data source. We also use the MSR SMS Toolkit [40], an existing SMS gateway solution from Microsoft Research India, to handle and respond to requests from users via a connection to an SMS sending/receiving port. The SMS toolkit is a simple programmable interface to SMS messaging, with hooks to applications that enable integration with various systems. A smartphone communicating with the processes running on the PC using the toolkit serves as the SMS sending and receiving port. Meanwhile, the phone can also be used for making and receiving calls.

All transaction requests involve three SMS messages per scheduled visit. First, the user requests their data to be delivered, along with their estimated time of arrival. A request can have different levels of priority, which would come at different costs, and determine how soon the request would be serviced, as well as the amount of resources allocated to the user's data. The system responds with when the request could be serviced and the cost of the service, asking the user to confirm the transaction within a deadline. This allows the user to gauge the freshness of their data as well as the cost. For example, it might be more valuable for a user to get their data in an hour rather than tomorrow, even at half the cost. Upon confirmation from the user, the system schedules the request for service. A simple SMS session might proceed as follows:

- user data request
  - `<fetch 30 minutes urgent>`
- system confirmation
  - `<Data could be ready in 30 minutes.`
    `Cost 4.5 Birr.  Confirm?>`
- user confirmation
  - `<Yes>`

3.1 Data Channels

Data channels provide an easy way to deliver personal data to the user through our framework, and represent how our system integrates into the existing infrastructure. These are often simple modifications to existing applications enabling them to use the Sulula API. To demonstrate how Sulula supports the plugin architecture and provide working examples of how channels can be effectively used to aggregate and consume personal data, we have implemented three channels on top of our system: an email channel, a news feed channel and a simple HTTP channel. These channels show how our system can be easily integrated to the existing infrastructure and incrementally support various data sources.

3.1.1 Email channel

Email represents the majority of private data that is consumed in developing countries [31]. This channel supports email communication using the most common mail protocols. We modified the PyMailGUI email client [23], a lightweight email application written in Python/Tk, to support our infrastructure. The client side plugin allows for secure registration and displaying of email messages, while the data source plugin interacts with service providers to fetch and send email. The data source plugin uses the owner's credentials to authenticate and fetch mail from a POP3 or IMAP server and send mail using SMTP much like a desktop mail client would. The basic difference between using the email channel and an unmodified desktop email client is that in our system, email administration and display occur at the data consumption end whereas email is fetched and sent at the data aggregator, with the two ends communicating using the Sulula infrastructure. To accomplish this, we devide the application to two parts as shown in figure 2. By employing an asynchronous data transfer mechanism provided through our system, the email channel provides a much more pleasant experience of email communication in challenged environments. Our modifications took a total of about 130 lines of Python code to implement.

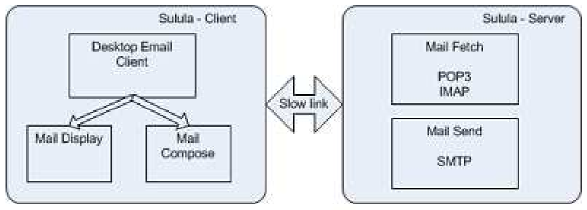

Figure 2: Email Channel

3.1.2 Newsfeed channel

This channel is used to aggregate news feeds in any of the common forms used on the web today, including RSS, RDF and Atom. We modified the NewsFeed [7] news aggregator to use our infrastructure for transferring data in a challenged network environment. As shown in Figure 3, the application is broken to two parts, with the aggregator running on the data source and the news displayer running on the data consumption end. As described in the example given in section 2, a user can register any news feed that she is interested in using the client plugin, which gets transported to the data aggregator. The newsfeed channel also supports authenticated feeds, which allows users to supply their credentials for accessing private news feeds. News is fetched asynchronously when the system gets a request for the user's data. We needed less than 50 lines of code to modify this application to use our system.

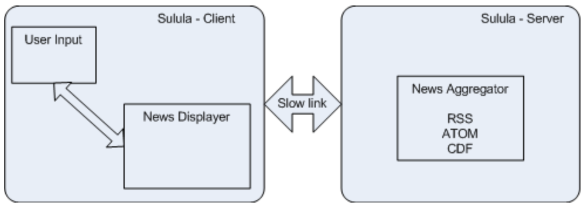

Figure 3: Newsfeed in Sulula

3.1.3 Simple HTML Channel

This channel provides data aggregation and display for simple HTTP/HTTPS pages. Studies of web access patterns [4, 8] show that there is a long tail of niche destinations that are often requested by a single user or a handful of users at most. This channel is useful for handling such requests that stand to gain little to nothing from conventional caching mechanisms. The client plugin allows users to input webpages they would like to be fetched ahead of time. These could be web portals, educational sites, fast updating news network sites etc. The data aggregator side plugin uses the request to download and package an HTTP/HTTPS website and readies it for delivery. This plugin can also be used as a building block for more sophisticated channels.

4. EVALUATION AND DISCUSSION

An ICTD researcher working in Uganda recently wrote about her experience using the Internet there, saying [17]:

> At the moment, I've totally given up on using my 64/64 WiMax+VSAT link via Infocom (which costs $300/month) and I'm using my Warid Telecom GPRS/EDGE modem (cost $60 + $40/month), which incidentally also claims speeds up to 128kbps (16KB/s), but in reality usually sits at about 2-5 KB/s on a good day (I am getting about 1.0KB/s now). The MTN EDGE/HSDPA service ($150 + $45 modem) is a bit of a joke and I have never seen it go above 1-2 KB/s (It's supposed to be 384kbps, or 48KB/s) ... I can't send emails and every time I manage to load my credit card website it times out and kicks me back to the log on page ...

Our main goal in building the Sulula infrastructure is relieving this kind of frustration, which is all too common in developing countries. In this section, we will discuss how our system improves user experience and adds value in these environments.

To evaluate our system, we used a client computer with a Windows Vista operating system running on an Intel Core 2 Duo processor, connected to a phone running Windows Mobile 5 that can send and receive SMS messages. Our data aggregator runs the Mac OS X Version 10.5.5 operating system, also with a Core 2 Duo processor. To simulate a challenged network environment, we used a WAN simulator [36] to cap the client's bandwidth at 15Kbps and introduce a latency of 500ms [1] as well as a 5% packet loss rate. These are reasonable numbers that a user might expect when going to an internet kiosk, and often better than our experience at several internet kiosks in Addis Ababa, Ethiopia.

Our workload consisted of checking email from a common service provider, Gmail, where there are a total of 5 new email messages in the users inbox, resulting in 1 MB of data transfer. We base our numbers on a recent study of email usage patterns [35]. The user also sends out one email with an attachment of a 200KB file. In total, the user has about 1.2MB of useful data transfer in the email session. In addition, we include one webpage and one blog that publishes a news feed, for a combined data transfer 500KB, to our hypothetical user's routine. We summarize the setup in table 2. Based on this setup, we evaluated our system from two perspectives, cost of using the system, including SMS messages, and also in terms of time savings realized.

Table 2: Experimental Setup

Data size		Bandwidth	Latency
Incoming Email	1MB	15Kbps	500ms
Outgoing Email	200KB		
Web & Newsfeed	500KB		

4.1 Performance

The biggest gain from using our system is significant time savings. Users spend a lot of time waiting in line, as well as slowly accomplishing tasks on the internet. When we ran our scenarrio in an unmodified kiosk environement, in addition to taking 33 minutes on average to complete, we had to refresh the page and restart downloads a few times because the page would simply hang. Sulula significantly cuts this wasted time by making a user's data available and ready before the user gets to the kiosk, as well as asynchronously transferring user generated data back to the data source. Data access from storage is almost instantaneous, and the only on-demand transaction is retrieving the encryption key from the server, which takes less than a minute. Going back to our user, Elsa, in Addis Ababa, this could mean being able to effectively use her time during the day, and freeing up the evenings for other important activities.

Table 3: Time Savings

Avg. Network Interactive Time	
Existing Kiosks	33 Minutes
Sulula	< 1 Minute

4.2 Cost and Pricing

As it stands now, most kiosks in developing countries charge their users by the minute spent connected to the internet. The average price from our visit to Ethiopia was around 0.4 Birr (about US $0.04) per minute. This is slightly higher than the cost of a single SMS message (0.3 Birr) as provided by the Ethiopian Telecommunication Corporation (ETC), which is the only service provider in the country.

A user is charged the per-minute cost regardless of the actual bandwidth delivered. So, if one spends about an hour trying to accomplish a task that is taking much longer than it should, it will cost about 24 Birr all together. Instead of this, our system encourages charging users based on the amount of data transferred. We believe this is fair to the user because internet kiosks often have multiple users share a single dialup connection, making any of the bandwidth promises irrelevant. Because this also encourages transparency, it would promote business among users who feel like they are paying for undelivered service. Users are also likely to be willing to pay more for such a service because of the significant time savings realized. Finally, the kiosks are free to set up their own prices as they see fit, and let the market determine who has the best value. To facilitate this approach, our system logs transaction information such as data size and date, and makes it available to users and kiosk owners.

The three SMS messages per transaction cost a total of 0.9 Birr. If a kiosk charged 5 cents (0.05 Birr) per 10KB of data transfer, our experimental scenario would cost the user a total of 9.5 Birr, including all SMS messages. This price point is hypothetical, and could be different in the market.

On the other hand, running our scenario in an unmodified kiosk environment on average takes 33 minutes to complete under the network settings we describe, which translates to 13.2 Birr. Even if kiosks were going to adjust their prices to reflect how much they could have made without using the system, our approach has a more tangible way of measuring the service provided, and by extension its value. This means, in our scenario, kiosks could raise their price as high as 7.1 cents per 10KB, and still not charge the customers any higher than what they already pay, but at a much higher level of satisfaction. In addition, the kiosk can provide different tiers of service based on how soon users want their data, and prioritizing requests accordingly.

4.3 Developing a digital marketplace

Network resources in developing countries are limited. However, we can work within a few constraints to increase value both to consumers and service providers. For example, the user might have some preferences such as the location of a kiosk in a certain part of town that does not require taking the city bus or the availability of additional services at a kiosk. In addition, different users will have different time frames and urgency regarding when they want their data available. On the other hand, kiosks can better manage their resources as well. For example, there could be discounts for orders placed in advance while charging more for short notices, peak hours etc. In addition, overnight fetched bits could be cheaper, when demand for the kiosk is low, and bandwidth is generally cheaper.

We believe our system could provide the foundation for building a market exchange for digital services. Sulula, coupled with a way to make mobile payments such as M-PESA [26], can foster a location and bidding based pricing system where a user can offer multiple kiosks with prices, and upon hearing from a few kiosks, go with the most perceived value. Kiosks could set their policy and prices however they see profitable, and adjust according to the market. Going further, this could also be done by a separate brokerage system that matches users and providers using a bidding process. While this could increase the complexity of the system and how users interact with it, a central service exchange will allow both users and kiosks to have more options for business.

4.4 Limitations

Basic limitations such as high latency in the network environment make delivering interactive data to the user, such as supporting an active browsing session, a bit tricky. One naive way to achieve interactivity in networks with very high latency would be prefetching huge amounts of data. However, this is not feasible since resources are limited to begin with. Nonetheless, applications can develop smarter mechanisms that leverage Sulula to provide an experience that resembles interactivity. For example, plugins could analyze usage history to selectively request what pieces of data to download, and provide a pseudo-interactive experience.

Another limitation in using the Sulula framework is the diversity of private data sources and the quirks in accessing them. For example, how a user authenticates to and consumes data from one social networking website could be different from how data is accessed from an otherwise similar service. One possible solution is implementing a data channel that learns the user's access behavior of various websites within a browser, and uses the information for accessing and presenting data in future sessions. In addition, the wide adoption of open source authentication and authorization techniques, such as OAuth [27] and OpenID [37], allows for more generic solutions.

5. TEST DEPLOYMENT

We ran a small in-country deployment of our system at a medium sized internet kiosk in the outskirts of Addis Ababa, Ethiopia. Our local partner has been operating the business for about a year, and provides a number of office services in the kiosk besides internet connectivity. In this section, we will describe our initial usage study, the telecom infrastructure in Ethiopia, and some details about our experience setting up and running the deployment.

5.1 Telecom in Ethiopia

The Ethiopian Telecommunication Corporation (ETC) is the sole provider of communication services in Ethiopia, spanning land lines, mobile phones and internet services. The latest statistical bulletin from ETC[11] shows Ethiopia has one of the lowest telecommunication penetration rates in Sub-Saharan Africa, with both internet and mobile phone subscribers less than 1 in 100. Mobile phone service was introduced in the country in 1999, and internet access only a few years earlier. In spite of this, Ethiopia has one of the highest mobile phone growth rates, with subscriptions increasing by more than 164% during the last two years of the bulletin, which is the third highest in Sub-Saharan Africa next to Nigeria and Angola, according to the ICT Index [18].

Internet subscription, on the other hand, has not been expanding nearly as fast, which can be attributed to the lack of infrastructure and the high cost when available. At the end of 2005, ETC reported less than 18,000 internet subscribers in a country of nearly 80 million. The number of internet users was estimated around 113, 000 and the number of personal computers around 225,000 during the same year. The cost of service ranges from 160 Birr setup and 60 Birr monthly charge for a 900 minute dial-up connection to 111,000 Birr setup and a monthly charge of 48,000 Birr (around $4500) for a 2MB leased connection [10].

These numbers paint a grim picture about the state of affairs regarding network resources in the country, which is also common in many other developing counties. Since the situation is likely to continue for the foreseeable future, systems like Sulula are helpful in improving the experience of people accessing the internet in these countries. As a result, Ethiopia was an ideal place for deploying our system.

5.2 Initial user study

At the beginning of the deployment, we interviewed 12 internet kiosk users about a range of issues regarding their internet usage patters and mobile phone access, along with some demographic background. We asked questions like how long they have been using the internet, how often they come to a kiosk, how they use the internet and what some of the inconveniences are. Table 4 summarizes a few of the responses.

Table 4: A few responses from initial user study

Average age	25
Average # of visits per week	2.2
Common visit time	late afternoon, evening
Average session length	44 mins
Common uses	Email, Sports, News, Chat
Avg # of email accounts	1.3

Some of the relevant takeaways from our study were:

- The average internet user is a young worker with at least a high school education
- Most internet users were also mobile subscribers
- Webmail was the most dominant use case for internet access
- Most access times overlap, usually during late afternoons and evenings

5.3 Deployment experience

5.3.1 *Language issues*

The national language of Ethiopia is Amharic, a Semitic language with its own unique alphabet. English is thought in school as a secondary language, allowing us simple communication with most kiosk customers. However, we needed to have local translation of documentations and interviews to make sure they were well understood. We made an Amharic video tutorial for the user-facing part of our system with the help of one of the co-authors, and provided translation help with all of the interviews and surveys.

5.3.2 *Shipping code and troubleshooting*

One of the main issues we faced in starting up our deployment was the difficulty of transferring program files and patches on demand. The poor network conditions make it prohibitively difficult to transfer big libraries and toolkits over the wire. We physically mailed the initial version our software, along with supporting libraries and tools, which took more than two weeks to arrive. After that, we had to adjust how often we updated our software and take careful notice about dependencies and additional requirements. For example, at one point, it was faster to modify the backend of our system to use a different database engine that was available in-country and upload the patch, rather than ship another version of the original database engine that was compatible with the specific OS installation at the kiosk.

Troubleshooting was another sticky point as many helpful tools such as remote desktop or video chat do not work well over the available network. As a result, we were limited to using instant messaging and low-resolution screenshots for solving some problems that came about during the initial phase of the deployment.

5.3.3 *Pilot run*

After the initial user study, we solicited four volunteers to test Sulula for two weeks. They were encouraged to use dummy email accounts and non-sensitive web content to avoid privacy issues during a testing phase. At the end of the two weeks, the users were asked to fill a survey describing their experience – giving us valuable feedback in making our system work better. The prominent pluses in the survey included the local language video tutorial that explained how to use the system, and the ability to multitask, allowing users to 'go about their business' while having data downloaded or uploaded at the kiosks.

In addition to fixing some bugs, we were also asked for a few feature improvements in our system. These include a better and intuitive user interface and a streamlined registration process that could possibly be shared among multiple kiosks. A more technical suggestion included the ability to describe how deep our HTML channel should go in downloading links on a specified page. This goes hand in hand with supporting pseudo-interactive sessions using Sulula, and we hope to consider it in future releases alongside some support for a digital marketplace.

6. RELATED WORK

There are a number of systems that use SMS for accessing advanced services provided over the network, allowing even basic handsets to do things like search [16] and community organizing [15]. Warana Unwired [39] looked at replacing an existing PC-based system for managing information in a sugarcane cooperative with an SMS-based mobile phone system in rural India. The solution enables farmers to get operational information such as sales and orders without having to make a trip to the central office. By sending an SMS messages with formatted text, the farmer would get back a response with the requested information, such as sugarcane output for the season. Our work is similar to these systems in using SMS as a bridge to services on the network. We build on the idea by leveraging the wide availability of mobile phones for delivering private data in challenged networks.

Analysis of WWW traffic in developing countries [8] has shown that the web traffic pattern is generally in-line with other studies elsewhere, following a Zipf-like distribution. A sizable portion of this traffic is personal in nature, ranging from webmail and educational websites to a large number of niche destinations that form the long tail of the distribution. After studying the traffic patterns for these countries, the authors also suggest some mechanisms that could help to improve performance of web access in those regions, such as client access methods for mail, converting software updates from polling to pushing, curbing irrelevant advertising and offline caching. We believe our work incorporates a number of their suggestions, and makes other performance enhancing techniques easier to implement.

Prefetching data has been widely studied [13, 14, 19, 20, 21, 22, 29] in both file system and web contexts. We extend this idea to private data in challenged network environments. For example, data staging [14] looked at opportunistically prefetching files and caching them on nearby surrogate machines to facilitate secure mobile data access. They designed the system to improve the performance of distributed file systems running on small, storage-limited pervasive computing devices. Although this work is similar to Sulula in prefetching content for ease of access later, our work is targeted at challenged network environments where bandwidth, rather than storage, is the limiting factor. In addtion, our work is adapted to usage patterns in developing regions.

HashCache [2], a configurable cache storage engine targeted at the 'next billion', runs at a very small memory footprint enabling multiple terabytes of external storage to be attached to relatively low-powered machines. While efficient cache storage such as HashCache reduces bandwidth consumption, and thereby the network connectivity expenses, we consider private or personal data where traditional caching mechanisms are not very useful.

Another area of web caching that has been studied well is delta-encoding [25, 34]. This approach looks at transmitting the difference between requested objects rather than whole objects. Although it requires cooperation from the server, this approach can be especially useful when dealing with requested objects that only change slowly. We believe our system provides a simple platform to implement delta-encoding without the cooperation of the data source as it can be incorporated with data aggregation points. In a similar fashion, value-based web caching [32], which uses the content of documents rather than URLs to make cache indexes, is straightforward to implement using Sulula.

Dittorent[33] looked at leveraging peer-to-peer technology to enable offline internet access at modem-speed dialup connection. They implement their solution using a browser plugin which looks up all file requests in the P2P network before downloading them from the internet. If the requested file is available in the network, it is downloaded at modem-speed from local peers. If it is not available, the file is downloaded from the source and then added to the P2P network for future use. However, this approach is unlikely to be useful for private data.

Other approaches for improving connectivity in challenged environments include optimizing high latency links [1] and delay tolerant networking [12]. The former suggests using simple QoS prioritization to provide more fine-grained, adaptive and behavioral classification of network flows rather than purely based on protocol and port, in order to optimize for latency for shorter flows whose performance depends more on latency. DTN proposes a network architecture and application interface structured around optionally-reliable asynchronous message forwarding, with limited expectations of end-to-end connectivity and node resources. Such an architecture can be used to achieve interoperability between networks deployed in mobile and extreme environments that often lack continues connectivity.

7. CONCLUSION

Traditional caching and prefetching techniques provide little to no assistance for personal data that is needed only by a single user. In challenged network environments, this means users often have to wait while their data is fetched from a remote server over a very slow link. We presented Sulula, an infrastructure for distributing private data in challenged network environments that leverages the near ubiquity of mobile phones for scheduling future requests. By using data channels, Sulula can easily integrate with the existing infrastructure and incrementally support various data sources. We implemented three data channels, and email channel, an news feed channel and an HTML channel, that use our framework to deliver private or personal data from common sources.

Our experimental results show Sulula can save users tens of minutes in a typical email/news reading session in challenged networks. In addition, our system facilitates a fair and transparent pricing structure that can improve how digital services are traded in developing regions. We also described a small, ongoing deployment in Addis Ababa, Ethiopia with early positive results. Sulula can be used alongside traditional caching and content distribution techniques to improve internet access experience in challenged network environments.

8. ACKNOWLEDGMENTS

This material is based upon work supported under a National Science Foundation Graduate Research Fellowship and NSF grant CNS-0509089. We would like to thank David Watson and Daniel Fabbri for reading and giving us feedback on the initial draft and our volunteers for testing our system. We also thank other members of the AutoMedia research group at the University of Michigan – Jason Flinn, T.J. Giuli, Brett Higgins and Timur Alperovich – for helping refine our ideas.

9. REFERENCES

[1] Y. Anokwa, C. Dixon, G. Borriello, and T. Parikh. Optimizing high latency links in the developing world. In *WiNS-DR '08*, pages 53–56, New York, NY, USA, 2008. ACM.

[2] A. Badam, K. Park, V. S. Pai, and L. L. Peterson. HashCache: cache storage for the next billion. In *NSDI'09*, pages 123–136, Berkeley, CA, USA, 2009. USENIX Association.

[3] C. Cook. XML-RPC for .NET. `www.xml-rpc.net`.

[4] C. Cunha, A. Bestavros, and M. Crovella. Characteristics of WWW Client-based Traces. Technical Report BUCS-TR-1995-010, Boston, MA, USA, 1995.

[5] The Digital Divide at a Glance, 2005. World Summit On the Information Society.

[6] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl. Globally Distributed Content Delivery. *IEEE Internet Computing*, 6(5):50–58, 2002.

[7] M. Doege. NewsFeed Aggregator. `home.arcor.de/mdoege/newsfeed/`.

[8] B. Du, M. Demmer, and E. Brewer. Analysis of WWW traffic in Cambodia and Ghana. In *WWW '06*, pages 771–780, New York, NY, USA, 2006. ACM.

[9] Leaders: The real digital divide - Technology and development. *The Economist*, 374:9–10, 2005.

[10] Ethiopia Tel. Co. Services, 2009. `www.telecom.net.et/services`.

[11] Annual Statistical Bulletin, 2005. Ethiopian Telecommunication Agency.

[12] K. Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM '03*, pages 27–34, New York, NY, USA, 2003. ACM.

[13] L. Fan, P. Cao, W. Lin, and Q. Jacobson. Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance. pages 178–187, 1999.

[14] J. Flinn, S. Sinnamohideen, N. Tolia, and M. Satyanaryanan. Data Staging on Untrusted Surrogates. In *FAST '03*, pages 15–28, Berkeley, CA, USA, 2003. USENIX Association.

[15] Frontline SMS. SMS Solution for NGO's. `www.frontlinesms.com`.

[16] Google. Google SMS. `www.google.com/sms`.

[17] M. Ho. The internet (or lack there of), 2009. www.ictdchick.com/blog.

[18] Measuring The Information Society, 2009. International Telecommunication Union.

[19] Z. Jiang and L. Kleinrock. Web Prefetching in a Mobile Environment. *IEEE Personal Communications*, 5:25–34, 1998.

[20] T. M. Kroeger, D. D. E. Long, and J. C. Mogul. Exploring the Bounds of Web Latency Reduction from Caching and Prefetching. In *USENIX Symposium on Internet Technologies and Systems*, 1997.

[21] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. pages 190–201, 2000.

[22] H. Lei and D. Duchamp. An Analytical Approach to File Prefetching. In *In Proceedings of the USENIX 1997 Annual Technical Conference*, pages 275–288, 1997.

[23] M. Lutz. PyMailGUI. `www.rmi.net/ lutz/`.

[24] K. W. Matthee, G. Mweemba, A. V. Pais, G. van Stam, and M. Rijken. Bringing Internet connectivity to rural Zambia using a collaborative approach. In *ICTD '07*, pages 47–58, 2007.

[25] J. C. Mogul, F. Douglis, A. Feldmann, and B. Krishnamurthy. Potential benefits of delta encoding and data compression for http. *SIGCOMM Comput. Commun. Rev.*, 27(4):181–194, 1997.

[26] O. Morawczynski and G. Miscione. Examining trust in mobile banking transactions in Kenya: The case of m-PESA in Kenya. *IFIP WG 9.4-University of Pretoria Joint Workshop*, 2008.

[27] OAuth Core Workgroup, OAuth Core 1.0, 2007. `http://oauth.net/core`.

[28] Operating System Market Share - Africa. `www.netapplications.com`.

[29] T. Palpanas and A. Mendelzon. Web Prefetching Using Partial Match Prediction. 1998.

[30] B. Petrazzini and M. Kibati. The Internet in developing countries. *Commun. ACM*, 42(6):31–36, 1999.

[31] A. Ratan and S. Bailur. Welfare, Agency and 'ICT for Development'. In *ICTD '07*, pages 119–130, 2007.

[32] S. C. Rhea, K. Liang, and E. Brewer. Value-based web caching. In *WWW '03*, pages 619–628, New York, NY, USA, 2003. ACM.

[33] U. Saif, A. Chudhary, S. Butt, N. Butt, and G. Murtaza. Internet for the Developing World: Offline Internet Access at Modem-speed Dialup Connections. In *ICTD '07*, pages 76–87, 2007.

[34] A. Savant and T. Suel. Server-friendly delta compression for efficient web access. pages 303–322, 2004.

[35] S. Shah and B. D. Noble. A study of e-mail patterns. *Softw. Pract. Exper.*, 37(14):1515–1538, 2007.

[36] Shunra. Desktop VE Wan Simulation. `www.shunra.com`.

[37] specs@openid.net. OpenID Authentication 2.0, 2007. `http://openid.net/developers/specs/`.

[38] K. Toyama, K. Kiri, M. L. Ratan, A. Nileshwar, R. Vedashree, and R. F. MacGregor. Rural kiosks in India. *MSR Techincal Report*, (MSR-TR-2004-146), 2004.

[39] R. Veeraraghavan, N. Yasodhar, and K. Toyama. Warana Unwired: Replacing PCs with Mobile Phones in a Rural Sugarcane Cooperative. *Information Technologies & International Development*, 5(1), 2009.

[40] V.Goyal and S.Blagsved. SMS server toolkit, 2007. `www.codeplex.com/smstoolkit`.

[41] D. Winer. XML-RPC Specification. `www.xmlrpc.com/spec`.