UjU: SMS-Based Applications Made Easy

Lu Wei-Chih† Matt Tierney‡ Jay Chen‡ Faiz Kazi‡ Alfredo Hubard*
Jesus Garcia Pasquel* Lakshminarayanan Subramanian‡ Bharat Rao†

†Polytechnic Institute of NYU ‡New York University *CAME

uju.news.cs.nyu.edu

ABSTRACT

A significant fraction of mobile users in the rural developing world use low-end mobile devices and have restricted data connectivity services due to a variety of economic factors. These devices have restricted capabilities with voice and SMS remaining the primary communication channels. The penetration of mobile information services in rural areas has largely been limited especially since all applications are operator controlled and very few applications have been adopted on a large scale.

This paper presents the design and implementation of UjU,[1] a mobile platform that enables users to develop new SMS-based mobile applications on top of a common platform. Given that the SMS channel is extremely constrained to 140 byte messages, UjU is designed to support database-centric applications that express and operate upon information in structured formats. In UjU, specifying a new application is equivalent to configuring an XML schema. Apart from exporting a standard set of operations, UjU allows the developer to specify new application-specific operations as XML forms. To make efficient use of the SMS channel, UjU supports a semantic compression engine that leverages the structured nature of the information transmitted. UjU includes a simple reliability layer to cope with message losses and uses a user-centric consistency model to handle data inconsistencies. We have configured and tested UjU for several SMS-based applications and describe our experiences in tailoring UjU to develop five real-world applications in the areas of mobile microfinance and mobile healthcare; four of them have been deployed in Ghana and Mexico.

[1]Uju is the Swahili word for "sent" as well as a shortening of the Swahili word for "message."

ACM DEV'10, December 17–18, 2010, London, United Kingdom.

1. INTRODUCTION

The massive penetration of cellular services in developing regions around the world has fueled the emergence of several development-centric mobile applications and services especially in the context of healthcare, education and microfinance. OpenXcode [25], OpenRosa [24], ODK [33], FoneAstra [11], FrontlineSMS [15], RapidSMS [28], OpenMRS [23],and Voxiva [31] are examples of a few successful projects that have explored the use of mobile phones as a low-cost computing platform for distributed mobile health applications.

One of the fundamental roadblocks to large-scale adoption of mobile applications and services has been that of *operational sustainability.* From the device standpoint, the software implementation of most mobile applications (except a few SMS-based applications) requires a high-end mobile device (such as a smartphone) and is typically too heavyweight for low-end mobile phones. In addition, these systems rely on SQL database implementations on standard TCP/IP networking stacks. These in turn depend on GPRS network connectivity which is feasible only in urban settings; in most rural settings only voice and Short Messaging Service (SMS) capabilities (140 byte packets) are available.

On the opposite end of the spectrum are lightweight applications like FrontlineSMS [15] and RapidSMS [28], which use the SMS channel and are designed for low-end mobile devices. These systems require very little infrastructure and low-end mobile phones are adequate. The main drawback of these systems is that forms and structured data do not influence the design of the messaging system. The result is feature-poor systems using ad-hoc messages and responses utilize the limited bandwidth inefficiently.

To address these limitations, we present the design and implementation of UjU, a mobile platform that enables users to develop new SMS-based mobile applications on top of a common platform. UjU is a generic, customizable platform that significantly simplifies the design of new SMS-based applications. Given that the SMS channel is extremely constrained with only 140 bytes available, most SMS-based applications leverage

a structured data format where the information transmitted over the wire is limited. UjU explicitly assumes the presence of an inherent data organization structure within each application and any data manipulation operation is treated as a simple fetch or update operation on structured data.

The design of UjU makes two important contributions. First, UjU significantly simplifies the design of new SMS based applications on top of a common platform. Users with no programming experience can potentially build new SMS based applications within a short time period. Building an application on the top of UjU is as easy as editing an XML file or using a web form. Second, to minimize the operational cost of SMS-based applications, UjU supports a semantic compression layer that minimizes the number of messages sent over the air. To enhance the compression levels, UjU allows users to aggregate several operations on to a single message. UjU also supports a lightweight reliability layer to handle message losses or reordering effects.

We have tested UjU across a wide-range of applications and have deployed different versions of the systems in Ghana and Mexico. Using a detailed evaluation, we show that UjU can: (a) reduce the time to create new apps; (b) achieve high compression levels; (c) support high levels of aggregation at both a record and operation level; (d) have a low overhead for providing reliability. Together, UjU can reduce the operational cost of applications. Moreover, it is possible configure a new application *Over-The-Air* using few SMS messages.

We also demonstrate the utility of UjU using five real-world applications: (a) a pharmacovigilance system deployed in Ghana; (b) a low-cost drug-tracking system developed for the Clinton Foundation; (c) a mobile microfinance application deployed in Mexico; (d) a customer service application for microfinance customers in Mexico; (e) an essential drug list query application for doctors, health workers and pharmacists which is a transliteration of a drug list book used in Ghana. The common takeaway message across all these applications is that UjU simplifies application design and reduces the operational cost of applications.

2. BACKGROUND AND RELATED WORK

In this section, we will first briefly describe motivate why SMS-based applications and then discuss important related work within the mobile application space.

2.1 Why SMS-based applications?

To motivate UjU, it is essential to understand why would users want to use SMS-based applications. According to ITU[18], the penetration rate of mobile cellular for developing regions is 49 per 100 inhabitants with roughly 246 million mobile subscriptions in Africa. However, the mobile usage is still low due to high usage costs with an average monthly cost of 23% of monthly gross national income per capital[18]. This is a significant cost given that a large fraction of the population earns less than US$2 per day. In addition, studies have shown that the mobile usage does not increase proportionally with increased income[27] suggesting that cost remains the main factor for low usage levels. The same study showed that mobile usage will significantly increase if the costs are brought substantially down[27]. The current mobile phone usage rates are also relatively high compared to public phone usage rates[27].

While usage remains currently very low, the low purchasing power also forces users to buy very cheap low-end mobile devices which cost roughly $20 per device[27]. Low-end devices come with very limited set of features with voice and SMS remaining the primary communication channels available from these devices. It is not clear whether smart phones will ever be able to break the $40 barrier and be available as a cheap communication device for the low-end markets. The current and projected price points of smart phones which support extended data services are significantly higher than the purchasing power of rural populations in developing regions. Also, the real need for these sophisticated devices is also far from established in developing regions.

Apart from basic communications, there has been a growing drive among mobile operators to enhance the range of mobile services in developing regions. SMS remains the primary data communication channel available to most end-users with low-end devices and hence is a popular platform for mobile applications. Data coverage has remained relatively low in many rural regions due to low user densities and low purchasing power. If voice usage is currently extremely constrained due to cost factors, the data services market is significantly lower than the voice market in these regions; this has remained a barrier for the adoption of data-connectivity oriented services.

SMS is universally available in all areas with cellular coverage and has been widely used in most developing regions. A recent work by Oliver [22, 21] showed how SMS can be used as a transport layer for building new mobile applications.

2.2 Sample SMS Applications

Mobile banking and healthcare are two popular areas where the need and drive for new applications has been immense. Every mobile operator in developing countries have their own platform for mobile banking mostly operating over SMS or USSD; the popular ones being M-Pesa [5], GCash [3] and mCheck [6].

At the recent Mobile Health summit [7], nearly 400 different mobile health applications in the context of developing regions were presented. SMS has been used as a transport for several mobile health applications.

FrontlineSMS [15] has been used as a platform for building SMS applications in more than 20 developing countries. The popularity of *FrontlineSMS* can be attributed to its ease of installation and use as well as its flexibility. Besides being used a data center, FrontlineSMS can be an SMS HUB to retransmit messages to a customized group. RapidSMS [28] is another widely used open-source platform that provides a environment for collecting data from a mobile device via SMS Text Message. There are some successful mobile health applications,such as child count [29], which have been built on the top of RapidSMS.

Several SMS-based counterfeit drug detection systems have been developed. mPedigree [8] is an ongoing effort geared specifically towards combating the problem of counterfeit drugs in the developing world (Ghana in particular). Epothecary[26] is another similar system which uses mobile camera phones for drug tracking.

Several major search engines have released their own SMS based search services. Google SMS service[17], Yahoo oneSearch[9], Chacha[2] and JustDial[4] are examples of SMS based web search services where the end user can SMS a search query and obtain a search response. SMSFind[12] is another recently developed SMS based search service.

2.3 Other Mobile Platforms

Outside of SMS based applications, there have been other popular non-SMS mobile application platforms in the context of development centric applications. We describe a few of these platforms. *openXcode* [25] is a free open-source software solution for collecting and management any kind of form based data on a mobile device. It allows users to fill out a form and send the information to a central control center from a mobile device. JavaRosa[19] is an open-source platform for collecting form base data as well. It supports XForms standard to create customized form for the user. Project GATHER [16] and CommCare [13] are built on the top of it. GATHER has developed a set of tools that can collect data from different device in the developing countries. CommCare aims to provide CHWs a platform that support home-based care and social support to HIV, tuberculosis and other chronic patients. OpenMRS is an open-source application which provides an extensible mobile medical record system. It has been adopted by Millennium Villages [20], FACES [14] and AMPATH [10]. Voxiva[31] is a commercial software mHealth solution that has been widely used. A common requirement across many of these platforms is the need for high-end mobile devices and the need for data connectivity to operate, both of which may not be ubiquitously available in rural settings.

3. SYSTEM OVERVIEW

In this section, we establish the goals of UjU. We then elaborate on how UjU achieves those goals through our design principles.

3.1 System Goals

We designed UjU around two goals:

1. For users without programming experience, make producing SMS-based applications simple.
2. Given the cost of an SMS message transmission and the 140 byte constraint of an SMS message, make SMS-based applications *operationally sustainable*.

Most users of SMS-based applications are not computer programmers; however, they are able to use a simple interface to create applications for their context. For instance, most mobile health efforts focus on transitioning from paper forms to electronic medical records, which can be accessed and modified by community health workers on the field using mobile devices. We designed UjU as a framework that can generate new applications with simple configuration from users without a programming background.

Operational sustainability is critical for any SMS-based application since sending and receiving SMS messages are expensive operations. Mobile phone-based projects are at the mercy of mobile operators to provide sufficient incentives for a large scale deployment; without these incentives, making a project sustainable has remains an elusive goal. To understand this better, consider a case in Malawi, where health workers use FrontlineSMS to update patient records via SMS messages. For context, the estimated HIV prevalence is 11.9% [30] of the 14 million person population [32], thereby accounting for 1.5 million HIV-infected patients in Malawi. The population covered by the mobile cellular network is 93% and one SMS message costs US$0.09. Assuming that the health workers send a single SMS message per month as an update per patient, the net communication cost of the system across all patients is US$135,000 per month. In the current program, a CHW visits five patients per day, requiring altogether 10,000 CHWs. The current salary of a CHW is around US$60. So, the total cost is US$(10000 * 60 + 135000)*12 = US$2.4M per year. The communication cost alone is high so incentives are essential to run a large scale program.

3.2 Design Principles

In this section, we illustrate how specific design principles of the UjU stack underpin the design of both the server-side and client-side components of our system.

Figure 1 illustrates the UjU design stack. Similar to conventional mobile application models, UjU assumes

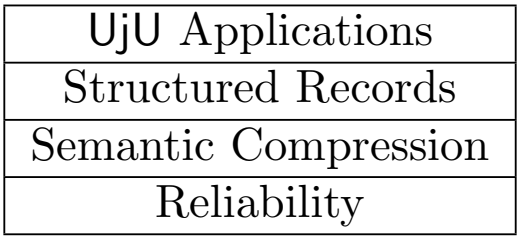

Figure 1: The UjU design stack.

a centralized server communicates with users who use applications on their mobile devices to fetch, update, and search records on the central server. We designed UjU for applications that operate over structured data, which allows us to leverage structure for more efficient SMS channel usage. Finally, we also implement lightweight mechanisms to handle the reliability challenges of SMS.

3.3 Guiding Principles

We now highlight the five design principles for the UjU stack.

1. UjU *Application Specification:* In UjU, specifying a new application is equivalent to configuring an XML file; the current interface allows users to create their forms using a simple web interface. Figure 2 presents the current version of the UjU web interface. UjU supports a standard set of data manipulation and search operations but also allows the user to create new operations using simple XML forms.
2. *Semantic Compression:* To make efficient use of the underlying SMS channel, UjU supports a semantic compression layer that semantically encodes a structured stream and achieves a high compression ratio in comparison to standard compression techniques.
3. *Aggregation of Operations and Records:* To enhance the channel utilization and minimize operational costs, UjU caches updates and aggregates multiple operations or records into a single session.
4. *Lightweight Reliability:* Given that the underlying SMS channel is unreliable, UjU supports a lightweight reliability layer that supports session-level reliability and uses delayed acknowledgments which are piggybacked in normal data messages.
5. *Consistency model:* Since intermittent updates may introduce data inconsistencies, UjU supports a flexible consistency model that allows administrators to handle inconsistencies. The default is an append-only model where every update is appended to a list and every fetch operation retrieves the last few updates from the append list.

Next, we elaborate on these five design principles.

3.3.1 Application Specification Made Easy

UjU is designed for supporting field level manipulations for structured records. UjU allows users to specify an application as an XML configuration file which basically captures the structure of records within an application and the parameters associated with each field. UjU supports two different type of XML configuration files, which are *Application Description and Form Description*. We elaborate on these next.

Application Description: An *Application Description* XML file is the primary application specification file that defines the structured record schema and their associated data types. Each field, in the schema, has three attributes: `Field ID`, `Name`, and `Range`. `Field ID` is a unique identify number, which will be referenced in the form description. `Name` of field will appear in the database and the screen of the client's mobile device. `Range` is a special field that is required for semantic compression. Figure 3(a) provides an example of the application description. In this particular example, the application id is "0" and its name is "CAME". To allow several applications to be continuously used on top of the same platform, every application is associated with an application ID. This application has only one database schema, which has 9 fields. To further simplify specification, one can derive the XML specification from a simple web form.

Form Description: The Form description defines a specific operation and the fields that relate to the operation. The best way to visualize an operation is a simple SQL statement as illustrated in Figure 3(a). Given any SQL operation, we can determine the corresponding form for that SQL statement. Figure 3(b) is the example form description for the SQL statement specified in Figure 3(c). In this form description, the id is **1**. This form belongs to a specified application with an identity number **0**. Every operator is associated with an identity and this operation has an id **3**.

UjU supports the following basic operations for any application: `Create`, `Update`, `Destroy`, `Fetch`, `Search`. The create and destroy operations allow users to create and delete records. The Fetch operation allows the user to fetch records that match a specific criterion. The simple fetch operation allows the users to pick a specific field and specify a constraint based on the field. The user interface also allows users to select multiple conditions and combine them using an AND or an OR clause. A user that needs a sophisticated fetch constraint needs to create a new XML Form description for the operation.

The Search operation is a field-agnostic fetch operation in UjU. The search model in UjU is fairly simplistic. Users can search for records based on a set of keywords where the search operation treats all fields as a text stream and outputs all records where the specified key-

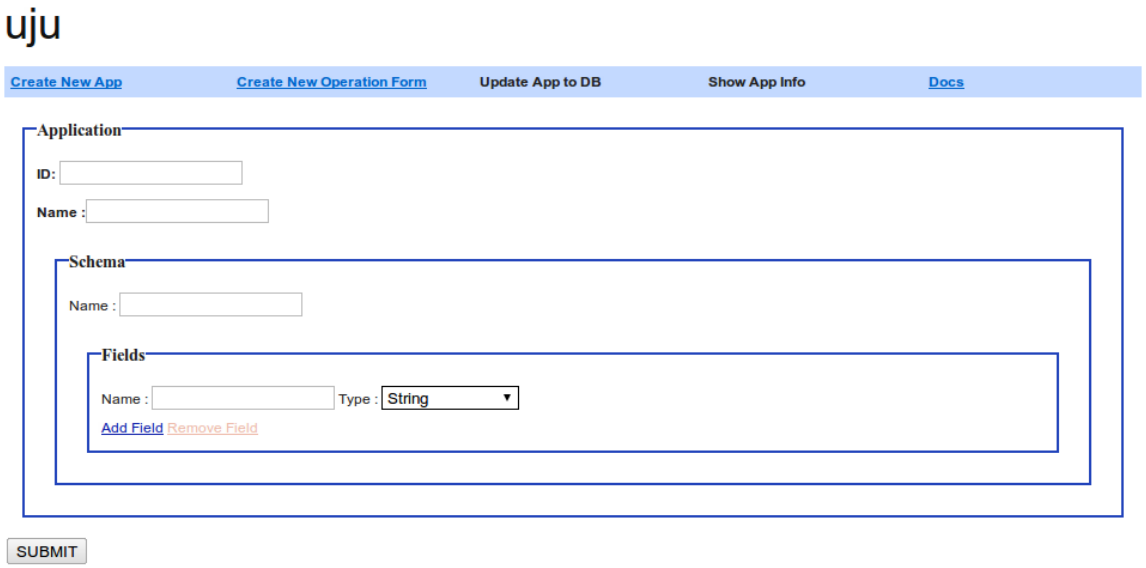

Figure 2: The current UjU web interface for creating an application description XML file.

words appear in some field in the record.

Updates and Consistency Model: The update operation allows the user to update one or more fields in a given record where we explicitly assume that a record is identified by a specific identifier known to the user or has a secondary key based on name or alternative parameters as a unique key. Given that users may access the database in an intermittent fashion, UjU uses a default append-only consistency model where all updates to a given record are appended to an append list. UjU explicitly exposes the inconsistency at the server to the administrator who is responsible for reconciling inconsistencies. Fields within UjU can be marked as either *one-time modify* or *append-only* or *last-modify*. The one-time modify fields are set during record creation (primary key values) and the append-only fields are not modified but maintained as an append-list. Users can also explicitly set the application or individual fields to last-modify which retains the last updated value independent of the order of updates. Append-lists are associated with a list of bounded length.

3.3.2 *Semantic Compression and Aggregation*

The goal of semantic compression is to minimize the number of messages that UjU uses for the various operations communicating with the server. Unlike traditional compression algorithms, the basic idea of semantic compression is to derive a compression codebook to represent each field (or a group of fields) in its lowest entropy using the smallest number of bits.

Consider a specific field and the universe of values that the field can take. A semantic compression codebook will generate a specific code for every possible value occupied by the field. Given the relative frequency of occurrence of every value in the universe, one can generate a Huffman code for the field which will guarantee that the field is represented using the lowest number of bits. For most applications, this frequency information is either not available or may change with time. An alternative simpler coding scheme is to used a fixed-length representation per field. If a field occupies n states, then the field can be represented using $\log n$ bits. If the universe of values for a field is unknown, we can adopt one of two strategies we simply associate the field with a fixed length (with no compression).

While this represents a relatively simple coded representation, we have also developed a joint-coding framework where the universe of fields is known or partially known. The basic idea is to measure the joint entropy of different fields and join multiple fields which are strongly correlated with each other. One can recursively apply the joint entropy till individual fields are not correlated with each other; i.e., joint entropy is comparable to the sum of the individual entropies. We do not adopt this joint entropy coding in the basic UjU codebase.

UjU simplifies the ability for the application to specify the universe associated with each field. UjU supports the following 5 basic data types: integer, string, single choice, multiple choice, set. Integers can be associated with a range value which automatically specifies the fixed-length code. Similarly string is a fixed-length code. Single choice is specified by $\log n$ bits for the n choices and multiple choice is represented using n bits. The set function allows users to specify a universe of values that can be used to derive the compression code. The universe is pre-specified in the application configuration file; hence the client and server can derive the

```
<?xml version="1.0" encoding="UTF-8" ?>
<application id="0" type="0" name="CAME">
 <schemas total="1">
  <schema id="0" name="client">
   <fields total="9" primary_field_id="0">
    <field id="0" name="Client ID" range="24"/>
    <field id="1" name="Group ID" range="16"/>
    <field id="2" name="Week's Number" range="8"/>
    <field id="3" name="Saving" range="18"/>
    <field id="4" name="CAME Capital" range="18"/>
    <field id="5" name="Interciclo Capital" range="18"/>
    <field id="6" name="Default" range="18"/>
    <field id="7" name="Late Payment Times" range="8"/>
    <field id="8" name="Miss Meeting Times" range="8"/>
   </fields>
  </schema>
 </schemas>
</application>
```

(a) Application Description XML file example.

```
<?xml version="1.0" encoding="UTF-8" ?>
<form id="1" application="0" operation="3" name="Client Info">
 <table>
  <schemas total="1">
   <schema id="0">
    <fields total="3" logical="AND" >
     <field id="0" operator="Equal to"/>
     <field id="1" operator="Equal to"/>
     <field id="2" operator="Less than"/>
    </fields>
    <fetch total="6">
     <field id="0"/>
     <field id="1"/>
     <field id="2"/>
     <field id="3"/>
     <field id="4"/>
     <field id="5"/>
    </fetch>
   </schema>
  </schemas>
 </table>
</form>
```

(b) Form Description XML file example.

```
SELECT ‘Client ID‘ ‘Group ID‘ ‘Week’s Number‘
‘Saving‘ ‘CAME Capital‘ ‘Interciclo Capital‘
FROM ‘CAME‘.‘client‘
WHERE ‘Client ID‘ = 485080
AND ‘Group ID‘ = 17291
AND ‘Week’s Number‘ < 5
```

(c) SQL statement corresponding to the example forms.

Figure 3: We present a unified example for the how to create a simple application with the components of UjU: application description, form description, and the generated SQL.

compression code directly.

To achieve better compression, UjU supports aggregation at both the operation level and the record level. To reduce the number of messages, the client maintains a queue of recent operations. When the queue has sufficient operations to fill a single SMS message, the client sends a batch of operations to the server. Similarly, multiple records can be compressed into a single message in the response from the server.

3.3.3 *Lightweight Reliability*

Since, SMS message delivery is not 100 percent reliable, nor are the messages that do arrive guaranteed to be in order [21]. Recent work by Oliver [22] has proposed the use of SMS as a transport channel. Our lightweight reliability is a simplified version of the transport presented by Oliver. Implementing a reliability model similar to TCP/IP would be excessively complex and expensive over SMS. Also, protocols that require multiple rounds of acknowledgements are out of the question since each message incurs a relatively high cost. To keep the cost as low as possible for the user, we propose a thin layer of reliability.

UjU uses an intermittent communication model between the mobile device and the server which is divided into *sessions*. A session is a two-way communication between the client and server. Most operations are client driven in UjU. During a session, the client sends a string of SMS messages to the server followed by a string of SMS messages in the reverse direction. Every session is associated with a 8-bit identity and a session can have up to 16 messages in each direction. Every message in either direction contains a 4-bit sequence number to indicate the message in the session. The 16 message limit is an artificial bound to simplify the design.

UjU uses three communication phases in a session. In the first phase, the client initials a session with a new session identifier and sends an aggregate stream of SMS messages. In the second phase, the server responds with the responses along with 16-bit bulk acknowledgment along with the responses to the client operations. If the response does not fit within 16 messages, the server needs to initiate a new session. All messages are semantically compressed to reduce the number of messages. If any message is lost or received out of order, the client or the server waits for a short period of time, before sending a NACK message with the unreceived messages. In the final phase after a round of interaction, the client can either immediately acknowledge receipt of the records to the server or can use lazy acknowledgements to provide an acknowledgement at the beginning of the next session. The lazy acknowledgement does

Figure 4: An example application written on J2ME for UjU.

save one additional SMS message at the client.

In general, the reliability layer is a batching based solution which does not guarantee perfect reliability but operates under the assumption that most SMS messages are correctly delivered. Under this protocol, a session may not complete only if : (a) all messages from client to server are lost; (b) all messages from server to client are lost.

4. IMPLEMENTATION

In this section, we describe some of the important implementation details of UjU. We elaborate on the nuances of our terminology as well as explore the client- and server-side details of UjU.

4.1 UjU Client

The primary role of UjU mobile phone client-side software is interacting with a remote database through SMS messages. Our client-side software is implemented as a J2ME application and has been tested across different Nokia phone models (Figure 4). UjU requires the server to specify a *User Data Header (UDH)* to enable the UjU client to receive the SMS message; if not specified, the mobile device will place the SMS message into the normal message inbox instead of passing the SMS message to the UjU client application. Considering the limitation of storage space on the mobile device, UjU stores only the critical information from the configuration file and does not store the entire XML file; this reduces the size of the configuration file and allows UjU to support many applications simultaneously. UjU uses the Record Management System (RMS) functionality in J2ME to maintain the configuration files and the individual local databases as row data. This allows UjU to store data in flash memory.

4.2 UjU Server

A deployed UjU server has the primary function of receiving SMS messages and, if necessary, sending response SMS messages.

UjU leverages Kannel, an open-source *Short Message Service Center Gateway*, or SMSC Gateway. It presents an HTTP and HTTPS request interface for applications to receive and send SMS messages. Depending on the type of SMSC, Kannel can use either AT commands to communicate with the GSM modem or a set of APIs/SDKs provided by the wireless carrier. Our current implementation uses a Samba 75 GSM Modem and Kannel uses AT commands to communicate with the modem.

The basic access model of UjU involves a client application running on the user's mobile phone. The client application accesses the remote database by sending and receiving SMS messages to and from the server's SMSC gateway.

The SMSC gateway will distribute the messages to UjU's server, which will first decode the messages. This decoding is necessary since most of the bits in the received SMS message are semantically compressed. Parseable header information in the SMS message indicates to the server which application description to use to semantically decompress the bits of the SMS. This application description information is fetched from the local "Application Description" database table and used to decompress the SMS message data. The final SQL query is made to the application's own database table based on the contents of the SMS message. If necessary, a return SMS message is prepared based on the original query and sent.

4.3 Creating Applications

Creating an application consists of two steps: (1) creating the **Application Description** and (2) creating a **Form Description** for each desired operation.

To clarify the discussion on supported operations, we present the following definitions. A *transactional operation* refers to operations that affect the stored state of an application; e.g., updating a record on the server is a transaction operation. Create, Update, Fetch, Destroy, and Count are already implemented transactional operations. A *logical operation* is a function whose input are data and whose output is a single value; e.g., the sum of fetched numeric values.

Currently, our system comes with built-in basic transactional operations for creating new form descriptions. For future work, our system will automatically allow users to define their own logical operations with the web interface.

A *applications* consists of a series of *input fields*. When an application writer specifies an application using our web interface, the web service automatically generates a downloadable XML file of the application description.

Currently, a user may fully specify their application description using our web interface.

5. EVALUATION

In this section, we present the evaluation of UjU across a set of sample SMS-based applications to illustrate the ease of application specification and the effectiveness of semantic compression. Specifically, we show that: 1) creating a simple application only takes 15 minutes and one can compress an application configuration file to be transmitted over-the-air in few SMS messages; and 2) UjU supports high levels of semantic compression and can aggregate several operations/records into a single SMS message.

5.1 Application Specification

5.1.1 Time to Create an App

We first evaluate the required time for creating an application on the top of UjU. We present evaluation based on a collection of form-based applications used in a variety of contexts, such as patient records, Craigslist, epidemiology. We list the time for creating applications in table 1. It only takes 15 minutes to create a very simple application for BMI records. This application only contains 10 fields, Patient ID, Sex, BMI value, etc. In addition, the configuration for the basic operations can also be automatically derived.

The creation time for an application is fully dependent on the number of fields and number of choices in the single or multi choice field. Taking Patient Intake Form as an example, there are only 20 fields in the application. However, there are 103 choices in this application. Thus, this application takes more than 55 minutes to create.

5.1.2 Size of Configuration file

	Application Description		Form Description	
	Client	Server	Client	Server
Patient Intake Form	1966	9075	376	2805
HIV/AIDS Diagnosis Form	1041	3990	338	2369
Used Car	359	1732	255	1963
Top 10 Digital Camera	924	4457	277	1886
Case Report	273	1425	203	1929
BMI	276	1372	200	1929

Table 2: Size of Configuration File (in bytes)

Considering the limited storage space on the mobile device, the size of configuration files should be as small as possible. Given that low-end mobile devices do not export a standard database interface, UjU uses *Record-Store Management System*, or RMS, on the client side. RMS is part of the J2ME environment and allows developers to record data in flash memory. A configuration file is represented by a single XML file. In the XML file, developer defines an *Element* and appends several attributes belonging to that element. An example of elements in the configuration file of an application might be:

```
<type id="1" name="integer" range="17"/>
<field id="0" name="Patient"
       type="1"/>
```

While this representation can potentially consume a lot of space, we can use the same semantic compression technique to represent an application configuration file as well as the operator-level configuration files. Table 2 shows the total size of all configuration files for different applications in bytes. The memory footprint of the configuration files are extremely small and most applications and operations can be transmitted to the client (Client configuration file) using a few SMS messages. While simple applications can be transmitted to the client in 2-3 SMS messages, complicated applications with several entries may consume up to 15 SMS messages. Individual operations can be transmitted to the client in less than 2-3 SMS messages. This allows for a powerful paradigm where SMS applications can be installed over the air using very few messages, which we will explore in future work.

5.2 Semantic Compression

To evaluate semantic compression, we consider the same six applications and compute the number of bits required for representing a single operation and a response. Creating/returning an entire record is the most space-consuming task for both the client and server. Table 3 shows the size of a record (in bits) after semantic compression. Across all these forms generated from different real-world settings, UjU can condense each form roughly 200-400 bits per record. In the worst case, for certain large forms, UjU may consume close to 600-700 bits to represent an entire record. We note that the same forms represented in text format can consume a few KB even after Gzip.

Such a compact representation allows UjU to aggregate multiple records in a single SMS message. In the best case, UjU is able to transmit 8 records into one single SMS message. This level for aggregation represents a lower bound since these are create operations where the entire record is transmitted over the wire.

Most common operations fetch or update very few fields within the existing records. Under such conditions, UjU can support much higher compression levels. In the case where users are interested in fetching only specific fields, UjU allows users to easily specify these fields using a simple selection interface on their mobile device. We evaluate this using the Patient Intake Form where we consider two different **fetch** operations will only fetch the patient important medical information or emergency contact information (along with basic patient information). Table 4 shows the result that UjU

	Number of Fields								Create Time (minutes)
	Integer	Date	Generic String	Name	Multiple Choice	Single Choice	Boolean	Total	
Patient Intake Form	19	2	12	6	0[0]	3[103]	0	40	55
HIV/AIDS Diagnosis Form	13	9	1	1	0[0]	5[15]	5	25	35
Used Car	5	1	1	1	1[12]	0[0]	1	9	18
Top 10 Digital Camera	2	1	1	0	0[0]	2[59]	0	6	30
Case Report	6	2	2	0	0[0]	1[2]	0	11	15
BMI	4	2	2	1	0[0]	1[2]	0	10	15

Table 1: Time for members of our lab to create an application on top of UjU.

	bits/record	record/message
Patient Intake Form	674	1
HIV/AIDS Diagnosis Form	188	5
Used Car	207	4
Top 10 Digital Camera	116	8
Case Report	231	4
BMI	262	3

Table 3: Required bit per record

	bits/record	record/message
Fetch Emergency Content	206	5
Fetch Patient Content	159	6

Table 4: Fetch Interesting Fields

can accommodate 5-6 records in a SMS message including the protocol header bits. Compared to fetching an entire record, the relative cost of this fetch decreases by a factor of 5.

What we have shown so far is the number of bits per record used by UjU. Most operations of UjU can be represented using much fewer bits. One can view an operation as a procedure call with parameters. In UjU, every application and every operation within the application has a unique ID which represents a few bits. Apart from these two, the parameters corresponding to a fetch operation is the list of positions with the corresponding search constraint. In practice, we find that most fetch and update operations can be represented in much lower than 100 bits.

In summary, semantic compression can achieve high compression ratios and provide high levels of operator and record level aggregation for a single SMS message.

6. REAL-WORLD APPLICATIONS

In this section, we present the details of the field deployment of UjU. We provide a detailed account of the utility of UjU in five different scenarios: i) a pharmacovigilance system in Ghana, ii) a low cost drug tracking system iii) a mobile microfinance application deployed in Mexico, iv) a drug list query application for doctors, pharmacists and health workers, and v) a customer service application for microfinance customers in Mexico.

6.1 Pharmacovigilance system

In Ghana, we worked with Korle-Bu Teaching Hospital to build a Pharmacovigilance system. The goal of this system was twofold: i) track the side effects of a drug among patients, ii) remind patients to take their medication on a timely basis. We briefly describe them below.

We divide around 100 drugs into 15 categories. For each category, we record the patient information, medication information such as dosage and schedule. Patients can fill out the form (on the phone) and update the database with the current information. Doctors have the option to fetch the records stored on the database according to the medication or side-effect.

The second application was a patient reminder system in Ghana which sends out a SMS text message to remind patients to take their medication according to the prescription. As dosage varies for each prescription, the patients needed to be reminded on a timely basis. Our system will automatically generate the text message according to the prescription, so that the patients are aware of both the time and medication that they need to consume.

We are currently working with Tigo Mobile, a mobile operator, to distribute the application on a larger scale. Tigo Mobile has volunteered to provide SIM cards to patients in this pilot study.

6.2 Drug List

Ghanaian doctors and health workers use a book titled Ghana Essential Medicines List which contains a list of commonly used drugs with usage, dosage information, and the level of care required for a particular ailment or disease. The pharmacists can check the usage of the medication on the prescription (by referring to Ghana Essential Medicines List) before they give the drug to patients. The goal of this deployment is to build a SMS based system that acts as a simple lookup service to Ghana Essential Medicines List and that is easy to use, for a pharmacist.

Ghana Essential Medicines List contains 29 different categories, of which some categories contain only few drugs. So, we divided the list into 14 categories and bundled up the rest of the (15) categories into a separate category. The system was designed to be a fetch-only application. The information in the database is not supposed to be modified by the user or client. Beside the information list in Ghana Essential Medicines List, we also put the potential side effects corresponding to

each drug. Thus, patient can easy fetch the drug information and check for symptoms of the side effects of the drug. This system has been developed for Korle-Bu Hospital.

6.3 Microfinance

CAME is a microfinance institution in Mexico City that primarily provides loans in rural areas [1]. CAME works on the concept of group lending. Group meeting is one of its main aspects, where group of people living in the nearby village gather on a weekly basis to either accept or return loans.

In this deployment, we developed an application to support the group meeting and help the CAME officers to judge the credibility of the loanee. To judge the credibility of the loanee, the CAME officer needs the credit information of the loanee for the past 16 weeks. The CAME officer used to rely on GPRS to fetch this information from the CAME servers. Our application stores and fetches the client information locally from the mobile device and does not communicate to a central server. The officer stores all the necessary information pertaining to a group on the mobile device and then leaves to the group meeting. This way, he can fetch the client information locally without receiving any data from a central server, which helps in reducing the data costs.

We also implemented two additional functionalities. The first set of operations concern the remote update of information. It provides following functionality, i) Fetching client information from server, ii) Updating the payment information to server for each client, iii) Creating a new record of Interciclo Loan, CAME credit. (Interciclo is the loan amount in a cycle, where each cycle is of 16 weeks)

The second set of operations deal with checking the loan/balance of each client and updating their records in the server. The set of operations is described as follows. *Closing the Cycle* procedure is executed at the last week of the cycle and this procedure: i) Checks the status of loan of each client and makes sure all clients clean up the loan; ii) Distributes the earning from the loan to each clients; iii) Renews the loan amount for the next cycle; iv) Allows clients to deposit or withdraw from the savings account; v) Updates the client information to the server.

Compared to the current system which is running on the Palm PDA, CAME officers found our application easy to use due to its simple user interface and the use of keypad on the low end mobile device. One officer felt more comfortable typing the amount from the mobile keyboard than using the touch screen. In the future, we plan to add the functionality of printing the receipt via a Bluetooth enabled mobile printer.

6.4 Customer Service

Most clients of CAME do not have Internet access. The only way they can communicate with CAME is by making a phone call which costs one peso per minute. Customers typically experience a long wait time to talk to the next available customer service representative, which exacerbates the situation.

To improve the quality of customer service, we developed a system based on the SMS gateway in UjU. The system can i) record clients feedback in the database, ii) update clients information, iii) return requested information to client. With UjU, clients' phone number becomes their primary means of contact and CAME can send information to their clients using SMS. The clients can also report to CAME via SMS.

6.5 Drug Tracking

We developed an application called SmartTrack on top of UjU to track the flow of high cost commodities such as antiretroviral drugs right from the supplier down to the final pharmaceutical retailer. This application was demonstrated to the Clinton Foundation supply chain group and we are hoping to roll out a version of this system in a small district setting with their collaboration.

Consider a simple supply chain for antiretroviral drugs comprising a hierarchy of suppliers, distributors and customers. In order to make the flow of drugs reliable and consistent along this chain, each link needs to be able to inspect, record and report on the commodities that pass through. SmartTrack accomplishes this using a platform that operates on a mobile phone with a built-in camera and allows that phone to read and interpret a product barcode and then compare that against a central database to validate the product and record its location. Here, the UjU database consists of the barcode reading, the mobile device number, time of update and the location of the mobile device.

Every drug bottle or carton is tagged with a barcode which uniquely identifies each product. Barcodes are typically supplied by the drug manufacturer and generated from a secret key that makes it hard to create counterfeit identifiers. The barcodes of every item received at a port of entry are stored in a database housed on a central server. Each intermediary point in the supply chain is managed by an agent who is equipped with a pre-registered, SmartTrack loaded, mobile phone which has a built-in camera to read the barcode information. Upon receipt of any goods, the agent takes a picture of the bar code and SmartTrack recognizes and registers the barcode. This information is recorded in the cell phone which acts as the local data store at each supply chain point.

SmartTrack is versatile and adaptable and from a supply chain standpoint, the system can be used in a

variety of ways, i) to monitor commodity flow in real time at the granularity levels of a single pill bottle; ii) to track inventory at the pharmacy outlet and provide early detection of potential stock-outs; iii) to alert pharmacists to drug expiry; iv) to check the authenticity of pill bottles in order to prevent counterfeiting.

7. CONCLUSION AND FUTURE WORK

In this paper, we have presented UjU, a mobile platform that enables users to develop new SMS-based mobile applications on top of a common platform. It has been designed around the context-specific goals of robustness, ease of use and cost-effectiveness, and the more general goals of efficiency, reliability and consistency. We have developed 5 different real-world applications on top of UjU, four of which have already been deployed. We believe that UjU fills a significant gap in service provision and makes it easy for non-programmers to build their own SMS-based applications. UjU can also help in getting past the dependence on mobile operators to deploy new mobile applications and services on a large scale.

In the future, we plan to explore several optimizations to the current design. Depending on the consistency level required for the application, a further optimization could be to send the delta between the cached version and the updated version of the record. We would also like to enable application writers to define their own logical operations using our web interface. Also, UjU currently only supports database search on the server side. However, we are currently working towards extending UjU to support web search via the APIs provided by standard search engines such as twitter, Google, etc. on the server side. Finally, the current implementation of UjU only supports installing and updating an application manually. However, the application installation can itself be supported over the air.

8. REFERENCES

[1] CAME: Dinero con sentido. http://www.came.org.mx.
[2] Chacha. http://www.chacha.com/.
[3] Gcash. http://www.gcash.com/.
[4] Justdial. http://www.justdial.com/.
[5] M-pesa. http://www.safaricom.co.ke/index.php?id=745.
[6] mcheck. http://main.mchek.com/.
[7] Mobile health summit. http://mhealthsummit.org/.
[8] Mpedigree. http://www.mpedigree.org/home/.
[9] Yahoo one search. http://mobile.yahoo.com/search.
[10] AMPATH. http://medicine.iupui.edu/kenya/hiv.aids.html.
[11] G. Chaudhri, R. Borriello and W. Thies. FoneAstra: Making Mobile Phones Smarter. In *NSDR*, 2010.
[12] J. Chen, L. Subramanian, and E. Brewer. Sms-based mobile web search for low-end phones. MobiCom, 2010.
[13] CommCare. http://www.dimagi.com/content/commcare.html.
[14] FACES. http://www.faces-kenya.org/.
[15] FrontlineSMS. http://www.frontlinesms.com/.
[16] Gather. http://www.gatherdata.org/.
[17] Google SMS. http://www.google.com/sms.
[18] International Telecommunication Union. http://www.itu.int/.
[19] JavaRosa. http://code.javarosa.org.
[20] Millennium Villages . http://www.millenniumvillages.org/.
[21] E. Oliver. Exploiting the Short Message Service as a Control Channel in Challenged Network Environments. In *Challenged networks*. ACM, 2008.
[22] E. Oliver. Characterizing the transport behaviour of the short message service. In *MobiSys '10*. ACM, 2010.
[23] OpenMRS. http://openmrs.org/wiki/OpenMRS.
[24] OpenRosa. http://www.openrosa.org/.
[25] openXcode. http://www.openxdata.org/Main/WebHome.
[26] M. Paik, J. Chen, and L. Subramanian. Epothecary: cost-effective drug pedigree tracking and authentication using mobile phones. ACM, 2009.
[27] M. Paik et al. The Case for SmartTrack. *ICTD*, 2009.
[28] RapidSMS. http://www.rapidsms.org/.
[29] RapidSMS Case Study. http://www.rapidsms.org/case-studies/.
[30] UNAIDS. http://www.unaids.org/.
[31] Voxiva. http://www.voxiva.com/platform.php.
[32] WHO. http://www.who.int/countries/mwi/en/.
[33] A. Yaw, H. Carl, B. Waylon, L. Adam, T. Clint, and B. Gaetano. Open data kit:building information services for developing regions. ICTD, 2010.