MySMS:
Connecting Developing Regions Though SMS

Brad Campbell, Ruibo Li, Anthony Poon, and David St. Hilaire

Department of Computer Science and Engineering
University of Washington, Seattle, WA

{campbl, ruiboli, anthop, davidjsh}@cs.washington.edu

Abstract. The limited access to information in developing regions hinders the growth and potential of developing regions. Limited infrastructure makes it difficult to resolve the limited access to information. By utilizing existing cell phone technology to bridge the gap between remote clients and database servers, the information gap can be lessened. Voice transport is not simple enough to be viable means of information exchange. However, another cell phone technology, SMS, provides a medium that is both simple and flexible enough to be viable. MySMS takes advantage of this medium by providing a framework using SMS as a transport layer to allow SQL queries, notifications, remote code execution and rapid application development.

Keywords: SMS, Android Platform, SQL

1 Introduction

The information age has brought a plethora of advances to the world which many countries capitalize upon to increase the productivity and wealth of their citizens. However, developing regions cannot easily take advantage of these technological developments because of limited infrastructure, poor technology penetration and a workforce with limited education. These limitations inhibit the rate at which information may be gathered and processed, which in turn limits the capability for developing regions to respond to new information.

However, there is an infrastructure in place which can be utilized to assist in the gathering and processing of information. This infrastructure exists in the cellular networks of developing regions. Although there is limited computer penetration and restricted Internet connectivity; cell phone usage is on the rise, and cell phone networks cover a far greater percentage of the populace in contrast to Internet access [8]. If information can be easily gathered and shared using this existing cell phone technology it would provide a significant boost to developing regions.

There are simple examples that demonstrate the strength of utilizing existing cell phone networks. However, there are far greater applications for cell phones which have not been explored. Reporting medical incidents is relatively basic and can simply be done with SMS and no additional software. On the other hand, more complicated applications such as crime reports, weather reports, rainfall prediction require a more complicated support system of a server, a database, and the ability to access both server and database. This cannot be done easily with SMS messages because of the complexity of the server/database interaction.

The solution for this is to develop applications, which run on a client phone in conjunction with a phone connected to a server. Unfortunate developing these applications can be difficult due to varying cell phone support, complicated access to servers via cell phones and a limited code base on which existing developers can base their applications. MySMS addresses all of these complications by providing a framework for developers.

MySMS's interface layer provides both convenience and functionality to developers allowing faster and simpler development. The interface provides simplified access to a SQL database, a widely accepted database framework, and support for multiple applications. By providing simplified access to an extensive powerful database, developers will have the ability to transmit, share, and move information at the fastest speeds available in the existing infrastructure.

2 Related Work

Utilizing the SMS capabilities of a cell phone is not an unexplored region of development. There has been considerable work done in the past which has focused on utilizing the SMS capabilities of a cell phone for various forms of information access. However, none of the previous work done covers the scope or the level of support for phones as MySMS.

FrontlineSMS provides an excellent example of a standalone tool designed to send and receive large quantities of SMS messages [1]. Frontline requires only a computer and an attached cell phone, and is particularly useful for applications such as surveys and mass notifications. However, because Frontline is a standalone tool, the types of applications which it can support are limited.

SMSLib, SMS Toolkit, and SMS Server Tools are three different projects which allow a developer to create their own SMS-enabled applications. These services allow a computer to control the sending and receiving of SMS messages through a complaint phone or GSM modem connected to the computer. SMS Server Tools also allows developers to specify code to execute when a text message is received [2]. SMS Toolkit also supports signaling code to execute on the receipt of messages, though it only can be used with Windows Mobile phones [4]. On the other hand, while SMSLib does not support signaling developer code when a message is received, it does provides the additional benefit of tracking sent and received messages in a database [7].

Warana Unwired, developed by Microsoft Research India, provides a great example of how SMS-enabled applications can empower residents of developing regions. Before the Warana Unwired project, farmers were required to visit a number of special locations to use computers which allowed them to access information such as crop output shipped to the cooperative, prices for fertilizer, or other account information. The Warana Unwired project replaced this system with one based on cell phones that provided farmers with the ability to access this information by sending special codes via SMS messages. It was hoped that not only would this provide farmers with better and more convenient access, but would also create a system that was much lower cost and more maintainable than the cooperatives previous wired infrastructure [9].

Fig. 1. A farmer is checking crop output, prices, and account information using the Warana Unwired project. MySMS seeks to help develop more applications such as this one to empower people around in the world in developing regions.
Source: Warana Unwired [12]

Another application, built by Tseng, et al., is designed to collect data through SMS-enabled sensors. Information, such as humidity, wind speed, temperature, and number of trapped pests, was collected by sensors in the field that would then send this information back to a centralized server via SMS messages. Tseng, et al. finds that SMS is a mature technology that can easily and reliably handle the needs of automated data acquisition [10].

Other tools and frameworks, such as GSM-CONTROL and SMS Reception Center, attempt to extend on the functionality of the basic libraries. GSM-CONTROL is a tool which focuses on controlling external hardware and other automated processes using SMS messages. It also allows access to a SQL database by allowing users to set certain messages to be interpreted as queries, but requires requires a GSM modem, rather than a regular cell phone, in order to receive SMS messages [7]. SMS Reception Center is for-profit software designed to send and receive SMS messages without human interaction by setting off certain actions based on when an SMS Message arrives, much like the SMS Toolkit mentioned earlier. Some of these actions can be SQL queries, while others can prompt the server to execute a specific program or send a return message [9].

3 Approach

MySMS builds on the work done by SMS Toolkit, SMSLib, and other frameworks by including additional functionality focused on allowing developers to quickly and easily prototype connected, SMS-enabled applications. In order to do so, MySMS adopts the client-server application model with emphasis on wide-ranging application support and a large amount of provided code. The client and server application model is a common and basic model for many applications. By adopting this model, MySMS makes application development easy and natural. MySMS is explicitly based around sending queries and returning tables of results from a SQL database, and this allows for a simple interface while retaining a large degree of flexibility.

A MySMS client is any entity which can send SMS messages to the MySMS server. Within this broad definition, MySMS supports two distinct types of clients. A non-smart client is any phone which can send and receive SMS messages. These are the phones which can interact with MySMS or any other SMS library by having the user type in queries in plain text. This is the vast majority of phones, and this allows MySMS to help solve the most immediate problems with realistic hardware requirements.

In cases where a graphical interface provides a substantial value over a text interface, MySMS also uniquely supports smart clients which can run the MySMS client service. The client service on the smart client provides a service local to the phone that allows applications running on the phone to communicate via SMS messages without needing to interact with the messages directly. A MySMS smart client also provides many additional features, such as message fragmentation, reliability, and enhanced message encoding, that are not possible on a dumb client. By providing these additional features as well as abstracting the SMS based communication, the MySMS client service enables developers to easily write connected and compelling graphical applications that are more user friendly than text-based clients.

While MySMS supports high-powered clients, only minimal requirements are necessary for MySMS servers. Any computer that can support a SQL database and can connect a phone via USB or serial port can also run MySMS and serve MySMS-based applications. No additional infrastructure, such as internet access, is necessary for MySMS, and this makes deploying MySMS to developing regions a practical solution.

The client and service model that MySMS implements provides a framework to guide developers as well as code for many common features that applications need. This approach is also evident on the reusable code base that MySMS provides. To assist developers in easily writing their applications, MySMS provides many read-made modules for common, application-specific functions, such as message encoding and login. By providing capable default modules, MySMS allows the developer to implement applications without a large amount of effort. Alternately, developers may choose to write their own modules to fully customize MySMS to their own application.

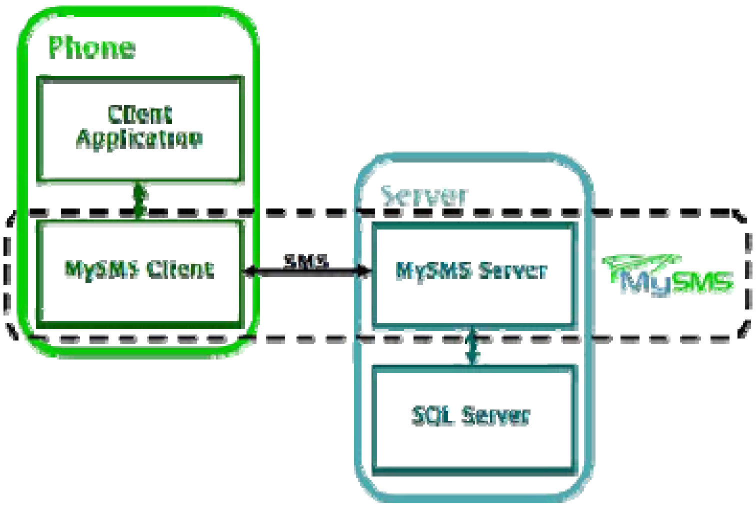


Fig. 2. The Server-Client model which MySMS utilizes. The client resides on the phone while the server side resides on a computer with an SQL server attached. The communication between client and server is done via SMS.

MySMS's approach focuses on providing a strong structural framework and a large supporting code base to make SMS application development easy. To effectively support many different applications, MySMS clients may utilize smart cellular phones running graphical applications or non-smart client phones using plain text SMS. MySMS's large code base can be reused for a variety of applications whether they be targeted toward smart phones, non-smart phones, or both. Overall the uniqueness of MySMS lies in the client-server application model, support for both text-based and graphical applications, and the reusable code base for easy application development.

MySMS utilizes a server-client model for handling SMS transactions and SQL queries. The engineering decision for this is based on the limited operating power of cell phones and the necessity for a server. The server acts as a central access point providing processing power, data access, and a point of control for reliability and security. The existence of the server allows MySMS to take advantage of the processing power and point of control for additional features which sets MySMS apart from previous projects.

Another advantage of the server-client model deployed by MySMS is related to the level of control the server has between the client and the SQL database. This allows for message processing and simplification of the requests on the client side with provided encoding and decoding modules. The purpose of this to allow support for both smart phones which may have GUI based applications and non-smart phones which may rely only on direct SMS text messages. By providing support for both smart and non-smart phones the chance for greater usage and application is higher.

To improve the usability and benefit of MySMS, the framework provides a great deal of usable code upon installation. This usable code is targeted specifically at developers interested in developing SMS powered applications on cell phones. Essential modules are encoding and decoding modules for the compressed transfer of SQL tables across SMS. Decoding modules for SQL result tables for pulling specific sets of information from a large query set.

All of this functionality is tied together with the server-client model. The server portion handles all the aforementioned features to support smart and non-smart phones and provide encode/decode modules for efficient data transfer and use. This particular model provides a great deal of reuse as well since developers can substitute any client or server module as long as the interaction is the same. That is to say, any client that communicates via SMS or any server which processes SMS messages can fit within this server-client system for basic functionality (e.g. SMS message processing, SMS inbound/outbound communication, server side code execution).

4 Implementation

MySMS is implemented with two separate pieces which work together via SMS. The SMS messages connect together the server side of MySMS and the client side. The server side handles queries, data acquisition, decoding, encoding, subscriptions, notifications, and database errors. The client side handles queries along with smart phone applications such as encoding, decoding, and phone specific applications. The interaction between the server side and client side can be simple or complicated depending on the platform of the phone. The engineering decisions made to implement certain features in a specific way will be explained as well.

The server portion of MySMS is implemented with a smart phone connected to a computer running a SQL database. The smart phone and computer need to have access to a reliable source of power because they are vulnerable to power outages which could cause serious disruptions in service, lost requests, and other service related implications.

The client portion of MySMS is designed to run on phones, which are smart with advanced features, or phones with limited features. The only limitation the client portion of MySMS may experience is delayed notifications in the case the phone cannot be contacted when the server sends a response to a query. It is the hope of MySMS that messages that cannot be delivered immediately will be queued at the service provider level and re-sent when the phone becomes available.

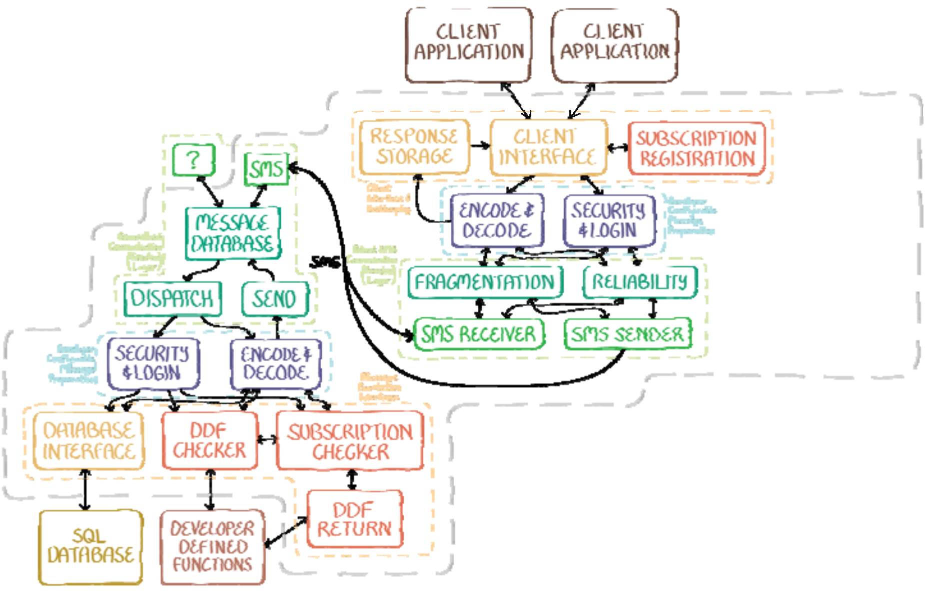


Fig. 3. The overall format of make up of MySMS with the server portions residing on the left side of the picture and the client portions residing on the right side. This is an expanded view of overall architecture of MySMS.

4.1 Server Implementation

The primary responsibility of the server portion of MySMS is to handle queries. The server utilizes the phone attached to a computer to receive queries, which are looked up in the database stored on the computer before responding to the query. To ensure that the transport level of MySMS is stable, a previous project known for good support and reliable transmission was chosen as the foundation for the server implementation of MySMS.

The implementation deployed uses a modified version of SMSLib's SMSServer. SMSLib's application is applicable for GSM compatible phones, which means certain smart phones are not supported by MySMS. However, the inbox/outbox system described separates the hardware of the server from SMSLib, providing the developer the ability to use any application, such as the MSRToolkit, to receive messages and insert them into the inbox of the database. This functionality greatly improves the overall usability of the MySMS tool and provides a great many more options, which is useful in any development environment. The reason for using SMSLib as the transmission engine for MySMS is because of the wide range of phones supported by SMSLib, hopefully providing developers with a wide range of options when deploying MySMS.

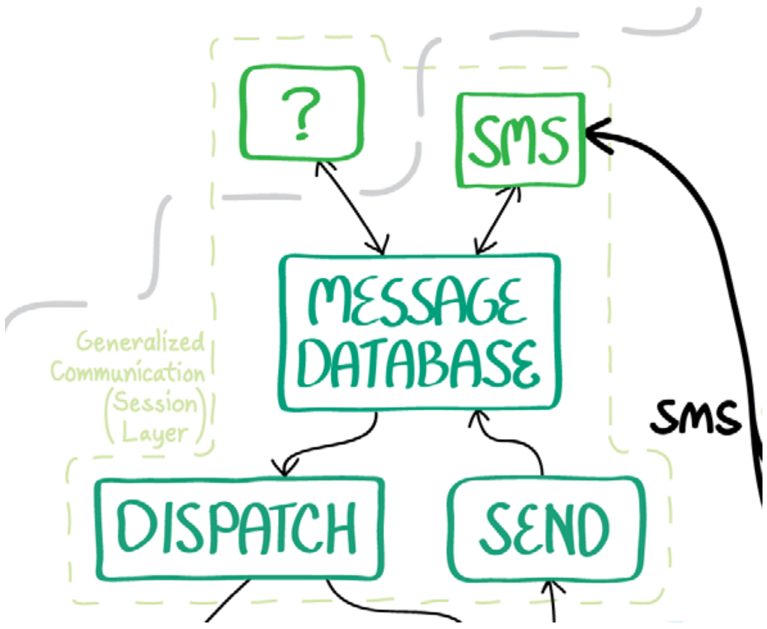


Fig. 4. This portion of the diagram shows the primary implementation with use of the message database described. The question mark signifies the ability for any application to add information to the message database, not only SMS Messages.

In order to properly use the SMSLib's transmission framework MySMS needed to follow the inbox/outbox SQL Table format to recieve and send SMS messages. This inbox/outbox is part of a SQL server hooked up to the receiving smart phone. Any messages the smart phone receives are placed into the inbox of the database once the header of the message or query matches. Any message or query that does not match the header is not saved into the inbox of the database. All outbound messages are placed in the outbox table, multiple SMS message are queued in the outbox and sent individually over SMS via the smart phone connected to the server. The outbound outbox on the server side is also transparent to the developer, thus the developer can potentially use any framework to do SMS transport.

4.2 Server Side SMS Message Decoding

Many of the benefits which MySMS provides developers exists in the ability for MySMS to sit between the inbound and outbound portions of the SMS transport, giving developers the ability to process the SMS message, when otherwise it would simply be treated as a raw SQL query or message. This is done by building on the inbox/outbox system which SMSLib deploys for SMS transport. When a message arrives in the inbox, MySMS is informed and pulls the message from the inbox. At this point the developer has numerous options on what is done with the message itself. If the message is an SQL query already properly formatted and ready to be queried to the database, the developer can pass it on and query the database immediately. If the message is in an encoded format which MySMS already provides a decoder for, the developer can pass the message to the decoder and MySMS will provide the developer an easy way to pull various parts of the message. Finally if the message is an encoded format only specific to the developer's own functionality, a separate encoder that the developer defined can be used to decode the message. For example the code below is one of the basic decoders provided by MySMS, it converts a given message which is a series of bytes into a string for easy processing.

```
public String decode(ByteBuffer buffer) throws
  IOException, ClassNotFoundException {

  ByteArrayInputStream ba_in = new
  ByteArrayInputStream(buffer.array());
  GZIPInputStream gz_in = new GZIPInputStream(ba_in);
  ObjectInputStream in = new ObjectInputStream(gz_in);
  String query = (String)in.readObject();
  in.close();
  return query;
}
```

In typical use case either the second or third option will be utlized because it is rare that a user will SMS an exact SQL query to be processed. This is mostly because it is inefficient because of all the unnecessary keywords ans spaces which SQL queries have, also it makes the SMS message itself abstract and difficult to understand for anyone but the developers themselves which would make it almostly completely inaccessible to users without any prior knowledge in SQL. For time savings, typically a developer would use the second option and utilize preexisting MySMS decoders for their messages, however since there are only a limited few decoders it means that more often than not if the developer has some special encoding they will have to write their own decoder. This can take up precious time which would be otherwise be used for adding features, the impact on development is covered in the evaluation portion of the paper. As a result, MySMS could be greatly expanded to include more decoders which developers could use, offering a greater variety would shorten development time and make MySMS an even more appealing framework.

4.3 Additional Server Features

Since MySMS does both the decoding and encoding at the server level, it allows the modules that run the server to do more than encode and decode SQL queries. This benefit allows MySMS to enable additional features that would otherwise not be available in the standard SQL command set such as subscriptions. The typical use case for a subscription is when a user or developer wants to receive a certain set of information at certain intervals. This can be accomplished by sending a special keyword that the server module recognizes. In this case MySMS uses the keyword "SUBSCRIBE". When the server sees the SQL query the parser which comes after the decoder will find this keyword and perform the appropriate action by making a note in the database that a specific phone number requested a subscription.

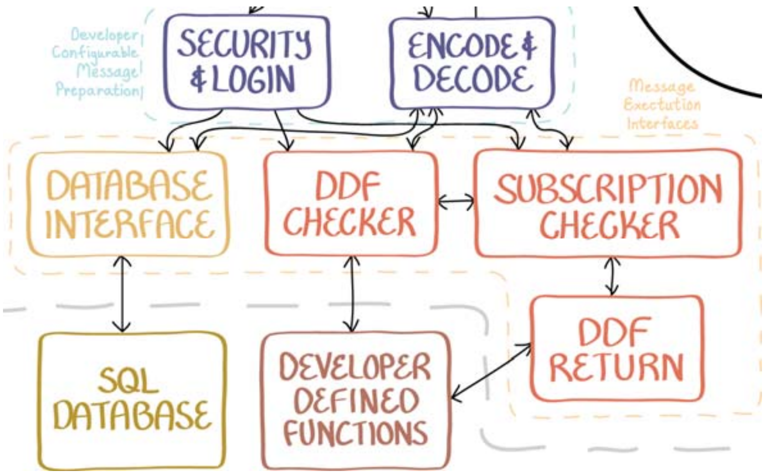


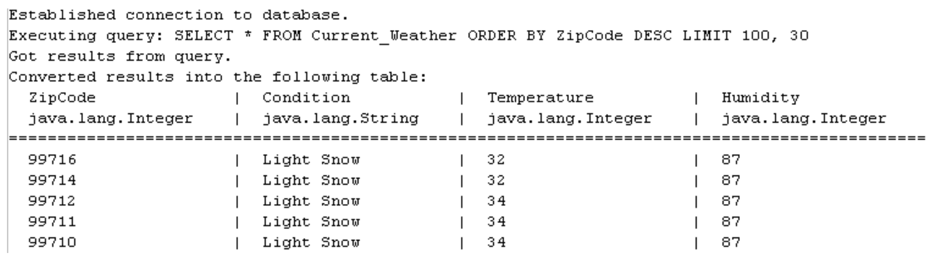
Fig. 5. These are additional features supported in MySMS and are handled between dispatch and send as shown in Fig. 4.

The mechanics of the subscription are simple because the underlying framework is not dependant upon anything else. Which means an additional module, which sits next to the server module and checks the subscriptions table for subscribers allows this feature to be fully implemented. Now whenever the server module has time it will scan through the subscription table and build the necessary outgoing packets and place them in the inbox table mentioned earlier. From there the SMSLib sending protocols will handle the details of sending the information to whoever requested it.

Another feature added to MySMS due to the ability of the server to parse and process messages are developer-defined functions. These functions allow cell phones that are incapable of performing complicated functions to send them as a request off to the server to be performed there. This is done the same way subscriptions are handled by decoding the message and looking for keywords. When a match is found the message is passed to the appropriate module, in this case the execution module for developer defined functions, where the functions are calculated and then placed back into the inbox table. Once a message is in the inbox table, it will be automatically sent to the requester without any more guidance from MySMS, simplifying the process a great deal.

4.4 Server Side SQL Table Encoding

After the developer is ready to to make a query to the SQL database, typically returns come back in an unwieldy format with multiple lines and column descriptions and a great deal of unnecessary aesthetic elements.To improve the ability of developers to obtain the information they need easily MySMS comes with decoders and encoders for the result table which an SQL query returns. This allows developers to grab only specific fields which they might be interested and ignore the others which is a great time saving resource. Another option is to use the encoder provided on the server side to send the entire table of applicable data and then utilize the complementary decoder to rebuild the encoded result table into something usable. This is particular useful because typically result tables can be considerable in size, which would take up multiple SMS messages, however by using an inbuilt encoder MySMS provides, many tables can be reduced down to only one SMS message.



```
Established connection to database.
Executing query: SELECT * FROM Current_Weather ORDER BY ZipCode DESC LIMIT 100, 30
Got results from query.
Converted results into the following table:
  ZipCode              |  Condition          |  Temperature          |  Humidity
  java.lang.Integer    |  java.lang.String   |  java.lang.Integer    |  java.lang.Integer
=============================================================================================
  99716                |  Light Snow         |  32                   |  87
  99714                |  Light Snow         |  32                   |  87
  99712                |  Light Snow         |  34                   |  87
  99711                |  Light Snow         |  34                   |  87
  99710                |  Light Snow         |  34                   |  87
```

Fig. 6. This is the format of the result table once it has been encoded. It is a drastic improvement in readability, information compression, and ease of access to specific fields.

4.5 Client Implementation

The client level applications on the cell phone for MySMS varies dependant on, phone models. Simpler phones that are only capable of sending and receiving SMS messages will have limited interaction with the client side software that MySMS provides. Typical usage will be to directly send an SMS message to the server's phone number and wait for a response. However, for more complicated phones such as the Android platform, developed by Google, there is substantial software to link together MySMS with Android applications to ease the development process.

In order to demonstrate the usefulness of MySMS client level to developer applications, the Android Platform served as a test bed. MySMS includes a set of services and protocols to assist developers in interaction with MySMS from the point an SMS message is received by the phone to the moment is transmitted. The entire process is based on a service that resides on the Android phone called MySMS_ClientService. The ClientService offers extensive abilities, these abilities manages the transit of SMS Messages on an Android Platform capable phone.

MySMS makes use of many special features of the Android Platform provides. One of these special features allows SMS messages to be intercepted and processed by an application so the user never sees the message inbox of the phone. This is particularly useful when an application is written that needs to abstract the database interactions away from the user so that the application seemingly runs smoothly without the user ever knowing that there is constant communication between the phone and the server.

There are limitations to this ability because the Android Platform cannot properly differentiate which message is intended for what application. The result is, if there are additional SMS programs running which also intercept the SMS message it may result in the deletion of a message intended for MySMS. This problem cannot be resolved but can be mitigated by clearly informing the user of the necessary actions needed to be taken for ensure smooth operation.

Under typical conditions any SMS communication related to MySMS will follow a specific protocol. Once the message is received on the Android Platform the MySMS_ClientService will catch the message thus hiding it from the user's inbox. From there the message is processed and sent through a decoding system. At the end of the decoding and processing MySMS_ClientService will have a ResultTable which is absolutely identical to the one the server portion of MySMS sent.

At this point a handshake is initiated between the developer application and MySMS_ClientService. Since the MySMS service runs consistently on the Android Platform when it is registered any developer application will be able to link and connect to it. From there the developer application will provide a reference to itself to the MySMS service. The purpose of this reference is so that the MySMS service can inform the developer application when the ResultTable has arrived and it is waiting to be picked up by the developer application.

This is done with two separate functions. One function exists inside the MySMS service and another function will exist inside the developer's application. Upon the startup of the developer's application it will call the function inside the MySMS service to inform the service of the existence of a new developer application. The service will make note of this and remember to keep track of any messages directed toward this developer application. In turn, after the service has found a message directed at the developer application, the service will call the function inside the developer's application alerting the application to the awaiting ResultTable. Along with informing the application, an attached MessageID will accompany the call so that when the developer application calls the service to ask for the ResultTable the appropriate message is returned. This process streamlines the communication between the developer application and the MySMS service.

In the case where the developer application wishes to make a query the primary method is to make a request through the MySMS service. There are predefined functions written into the MySMS_ClientService that constantly run which are utilized to make these queries. In the case where the developer application wishes to send a query, the process will be to call the function inside the service and pass the necessary information. The service will abstract the other necessary steps such as encoding the message and placing the required headers onto the message before being sent to the MySMS server.

One aspect of the implantation involves coordination between the client and server – this aspect is the security portion of MySMS. Currently the implantation of security in MySMS is limited, however the framework is set up to establish a security system much akin to UNIX groups and users. The implementation on the server is based on user names, user names are attached to groups which have specific permissions to execute certain SQL queries, run certain DDFs, and subscribe and unsubscribe to certain data elements. Additional security is being worked on and is considered part of future work.

5 Evaluation

MySMS exists as a platform for developers to create applications easily and quickly utilize SMS messages for the transport of SQL queries. Since that is MySMS's primary feature, it is most important to ensure the stability, reliability and functionality of this feature. In addition to this primary feature there are a wide spectrum of additional features which makes MySMS more viable, useful, and essential in contrast to other toolkits. These features are decoding, encoding, fragmented messages, notifications, subscriptions, developer defined functions and security. All of these features need to be objectively evaluated for an objective and effective understanding on the progress and potential of the MySMS project.

5.1 Feature Evaluations

The primary feature of sending and receiving SMS messages for the purposes of SQL transactions have been tested a great deal. Current tests show little discrepancy in the success of the feature. As long as the client can successfully send the SQL query the server is capable of parsing that query and responding with the correct ResultTable in the case of a smart phone, in the case that the phone does not have the capabilities of a smart phone, the raw data is returned. Considering the limitations of both smart and limited phones, this feature has been developed to mature stage with no known bugs and high reliability. There is no indication that this feature is not working, nor is there any indication that edge cases such as oversized messages or broken queries would result in unexpected behavior from the MySMS server modules.

The success rate which SQL queries are being handled by the MySMS server would also suggest that the encoding and decoding are working successfully. Since each message from a smart phone is first encoded at the client side, decoded and re-encoded on the server side only to be decoded again at the client side when the information is returned; any abnormalities in the encoding or decoding process would cause the entire SQL query to fail. Such a failure would be immediately recognized though the problem would not be immediately isolated. After considerable testing and preparation, encoding and decoding work as expected, which in turn supports the primary feature of MySMS.

Typically SQL queries can be lengthy and request a great deal of information. It is not uncommon for an SQL query to return a ResultTable that is thousands of characters in length. To resolve this issue MySMS deploys a system of fragmenting messages so that large messages are broken down into smaller ones. Due to the success rate of the SQL queries and the tests related to these queries it is safe to say that the fragmentation and defragmentation is also working as expected. In the case where it were not, large messages would be garbled, broken, or simply make no sense at all once they were decoded.

The extended features of MySMS classified as notifications and subscriptions have just been incorporated into the primary code of the package. Initial tests show successful implementation though there are still pipeline questions regarding how the properly inform the client application when there is a standing notification or subscription awaiting in the case the application has been shut down. Although the features are implemented the details have not entirely been worked out so there is only limited success.

Another keystone feature that MySMS includes is the ability for developers to insert their own functions into the MySMS pipeline. These functions can then be executed on the server side before being returned to the client. Initial tests seem to point toward a successful implementation of this feature though there has not been any specific application built onto this feature. This means that even though it appears to work, certain levels of stress on the server or other unknown factors may cause the feature to break down. At this time there is no conclusive evidence supporting this so it is considered a successful implementation.

5.2 Server Setup Times

One of the most important features, which MySMS provides, is the ease for developers to utilize it. This means developers need to be able to quickly set MySMS up and begin to use it right away. With this in consideration a great deal of time was put into setup time of MySMS and to reduce unnecessary hassle. Overall installation was quick and painless, simply downloading and deploying the modules necessary – all are packaged together by MySMS, which takes roughly an hour to complete the setup. Additional time is needed if the setup requires something beyond the default setup, however this is dependant on what the developer wishes to do with MySMS.

In the case the developer wishes to do something more specific with MySMS, typically the time needed depended on the difficulty of the task. In the case where the developer wishes to simply use queries directly from client applications there is no setup time because MySMS already expects that. In the case where the application will send specific information which is encoded differently the developer needs to take a small amount of time to write the decoder for the information sent. For simple applications this can be anywhere from thirty minutes to an hour.

5.3 Example Evaluation Application

To demonstrate the feasibility of MySMS in deploying applications fast and painlessly for developers, our evaluation required the creation of some sample applications. A total of three sample applications were developed, the simplest serves as a suitable example because it is a representation of the ease at which developers can deploy applications on cell phones with only minor additions to the server installation.

The simplest application based on the MySMS framework developed reported the weather given a zip code. This was supported on both smart phones and non-smart phones – limited to sending SMS messages via the keypad. The first benefit of MySMS is apparent in that for both the smart phone and non-smart phone the user only needs to text the zip code to the appropriate phone number for a response.

The removal of having the user input an SQL query directly as a text message improves both usability but also the access which the application may be deployed. Most users do not have an understanding of SQL queries, which means any information they submit over SMS in the form of a query will not appear transparent at all. However, when MySMS packages the information and transmits it in a format that the user can understand but the server can then interpret, it dramatically increases usability.

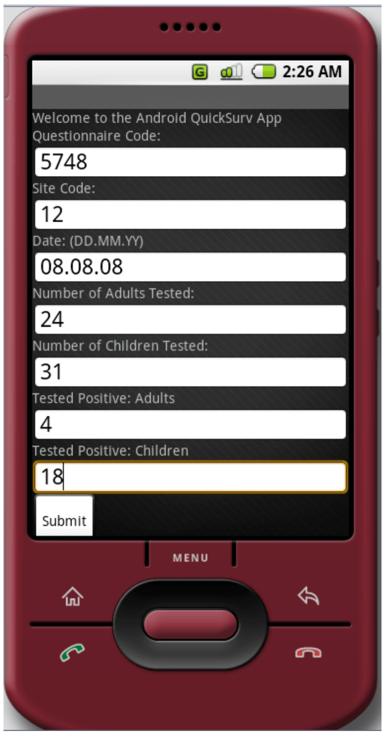
As a result of simplifying the message the user needs to send to the server the developer must provide a decoder module as mentioned earlier. This code is relatively painless and quick to write since the MySMS server side handles all of the transmission once the appropriate queries have been sent. Consider the following code:

```
public void InjectQuery(String str) {

  Statement cmd =
  con.createStatement(ResultSet.TYPE_FORWARD_ONLY,
  ResultSet.CONCUR_UPDATABLE);
  ResultSet rs = cmd.executeQuery("select * from
  smssvr_in where 'process' = 0");
  rs.close();
  cmd.close();
}
```

The example weather application is based primarily on this function which provides all of the decoding and server response. InjectQuery(String str) is called when an SMS message is received by the server and the message received matches the parameters set by the developer. The ease of development comes in that the message is passed straight through as a string for simple manipulations and processing. With this string the developer can create the appropriate SQL query, thus hiding it from the user who sent the SMS message originally. As a result, the user will only see a simple text request with a zip code.

Another example application based on existing work called QuickSurv[6] from the University of Washington is based primarily on the medical test result reporting. MySMS is perfect for situations where information needs to be reported back in a timely fashion so treatment may be administered before serious health issues arise. The benefit of using MySMS, aside from rapid development time, is the availability of a GUI to simply the data collection process and the encoding which is done in the background, meaning health workers do not have to manually type in the encoding on a cell phone keypad, thus reducing the chance of a mistake.



Welcome to the Android QuickSurv App

Questionnaire Code: 5748

Site Code: 12

Date: (DD.MM.YY) 08.08.08

Number of Adults Tested: 24

Number of Children Tested: 31

Tested Positive: Adults 4

Tested Positive: Children 18

Fig. 7. A graphical user interface for a QuickSurv application deployed on an Android smart phone. The text field represents the only information which the user needs to input before submitting the form.

Below is an example of the custom encoder used for the QuickSurv application. The message is encoded into the specified QuickSurv format by the client application and then when it is sent that message is encoded using our default QueryEncoder. In this code the message is first decoded using the default QueryDecoder and it is then split up so that the relevant information can be obtained. After the information is taken from the string a SQL query is created and returned.

```
public String decode(ByteBuffer buffer) throws Exception
  {
  QueryDecoder sqc = new SimpleQueryCoder();
  String encodedString = sqc.decode(buffer);

  String[] temp = encodedString.split("#");
  int questionaire_code = Integer.parseInt(temp[0]);
  int site_code = Integer.parseInt(temp[1]);

  temp = temp[2].split("\\*");
  String date = temp[0];
  int adults = Integer.parseInt(temp[1]);
  int children = Integer.parseInt(temp[3]);
  int adults_positive = Integer.parseInt(temp[4]);
  int children_positive = Integer.parseInt(temp[6]);

  String query = "INSERT INTO InputData
  (Questionaire_Code, Site_Code, Date, NumberAdults,
  NumberChildren,

  NumberAdultsPositive, NumberChildrenPositive,
  StocksOut, VTCOpen) VALUES ('" + questionaire_code +
  "', '" + site_code + "',

  '" + date + "', '" + adults + "', '" + children + "',
  '" + adults_positive + "', '" + children_positive +
  "');";

  return query;
}
```

5.4 Limitations and Obstacles

Although the overall progress of the project was smooth, there are a few discrepancies between what was the original intended behavior and the current behavior. The most significant discrepancy is found in the Android Platform on the client side. The best case scenario would be for the MySMS service to return a ResultTable to the querying application; the Android Platform has a documented issue that prevents the return of custom defined classes in conjunction with other custom classes. This means, the platform itself is incapable of carrying out the expected behavior, so it is not possible for MySMS to rectify this problem given the current Android SDK. The solution to this problem has been to return a String, a common non-custom class, to preserve the information requested.

A final note should be made on the limitations of the server due to network carrier constraints. During testing we found that the MySMS server was limited to sending one message per ten seconds. This issue was tracked and traced to carrier limitations. It seems that standard SMS messages are restricted to a single message every ten seconds. In this case if the server were to become overburdened by multiple messages it would significantly increase the delay between requests and responses. Although this is a serious issue for busy servers it can be solved by either working with the network carrier to remove the limit or increasing the throughput with a GSM modem or multiple phones.

In the case that the transmit limitation is removed MySMS has considerable throughput for handling queries since the time for message processing is nearly negligble. Most of the time taken to respond to a query is due to the limitation of sending SMS messages, without this limitation the throughput is restricted to the rate at which the inbox table is checked. Typically this number can be modified, but the default is around five seconds, thus every five seconds all messages in the inbox are gathered and processed. This allows for a throughput of thousands of messages every minute, the only restriction is limited to the processing power of the server itself.

6 Societal Implications

The introduction of MySMS to the global community is not dramatic such as the introduction of the steam engine or the internal combustion engine. Unlike a new technology which serves to simplify daily tasks, MySMS acts as an enabling technology which serves to improve the abilities of people to perform tasks which they were previously unable or faced significant obstacles to complete. Furthermore the effects of MySMS on the global community are limited to the willingness of developers to embrace this new framework.

The reasoning for this argument is the numerous projects which MySMS borrows and shares features with. These projects are mentioned above in related works and they serve both as solvency to prove that MySMS is a deployable and useful framework, while also demonstrating the slow and limited adaptation for developers to use SMS and cell phone technology. Although there have been numerous deployment of SMS and cell phone technology in these projects – all which serve some greater goal in bringing or sharing information, the adoption of these projects have been typically limited.

With these arguments in consideration it is not within reason to argue that the implications of MySMS are only limited to the applications which will be deployed with the framework. These applications can range anywhere between a simple weather reporting service to a complicated financial transport tool for those who need to make transactions without a computer. Depending on these applications the implications of MySMS could be enormous – especially given its early developmental stages.

The claim for the enormous implications of MySMS also is a result of the pioneering work done by MySMS in exploring the expanding features, which may be applied to SMS and cell phone technology. These features are unexplored and although may be considered safe they could open up entirely new security implications. Furthermore, like most pioneering work, following works may build upon MySMS, and if there is some considerable or unforeseen consequences, it may cause serious developmental problems in the future. Although the MySMS group has spend a great deal of time considering these issues, there is still the possibility for such an event.

7 Future Work

Currently there are limitations of scale related to MySMS. In the case where there are either too many applications, too many messages, or anything beyond the expected abilities of the server service will crawl to a stop. In addition there are still a great deal of other features which can be implemented to further the abilities of MySMS and make it a far more viable choice as a platform to develop cell phone based SMS applications.

Possible points of exploration for scaling can be considered by a closer inspection of the SMS Server Tools 3. This particular project managed to connect multiple devices up to a single computer. This means in the case where the smart phone the server is connected to cannot receive messages any faster multiple phone numbers could be employed. There is also potential for multiple applications and databases to be hosted on the same computer to make greater use of MySMS's capabilities.

Additional features that may be explored are simplifications to the SQL protocol so that the server can encode and decode shorter keywords for SQL queries. In addition, MySMS could implement a greater percentage of the SQL keywords. Currently only essential keywords are implemented, which means developers are limited to a subset of all the options which are available to them.

One prime feature that has the framework laid out but not the actual implementation is a greater level of security and encryption. Currently the implementation supports a system of encryption and security but it is not implemented. Expanding security features such as adding passwords, handshaking authentication, and phone number verification will ensure greater stability in MySMS and provide for a safer operating environment. In addition, encryption needs to be expanded so that messages can be securely protected against third parties and developers can pursue applications that carry sensitive information.

Another area of approach would be to expand MySMS to something beyond SQL database access. SMS messages can be used to retrieve many text-based sources of information such as e-mail, WebPages, and papers, and other text based information mediums. All of these mediums could be condensed down and eventually served by MySMS using the same protocols and server, except with a different method of accessing this information.

One important aspect of future work - which should be considered, is the increasing need to make SMS messages more efficient so that more information can be sent at once. One method of doing so is to introduce a form of smart encoding, which would shorten, and the overall message. This would allow for additional information to be sent over a single SMS message. An additional benefit to this line of development would be the increasing simplicity for non-smart phones to send queries. Often SQL queries can be very long and complicated, if a system of keywords were developed it would make queries much shorter.

8 Conclusion

At the beginning of this project, the MySMS team recognized both the potential of cellular phone networks; data transmission via SMS, and the essential role a database plays in the prorogation of data. The hopes of creating a framework for developers so that they may easily tap into this potential are realized in MySMS. Although the project itself is still in its initial stages, there has been great progress and a great deal of development already with many of the primary features and expectations met.

Although there are many other cellular phone SMS toolkits in existence, MySMS works to extend those abilities and bring new functionality to developers to further the rate of development for innovation and the prospering of ideas. Continual work on MySMS will reap greater benefits as more potential is explored and harvested bringing new advantages to developers who use not only MySMS but the idea MySMS is based on. This idea is simple, information empowers people, helps them grow, and brings out even greater potential; people should not have to wait for computers and fiber optic networks, that can be realized now with an innovative system of information collection and sharing utilizing a cell phone's SMS network.

Acknowledgements
The MySMS team would like to acknowledge our instructors and guidance advisors who kept us on track and made helpful suggestions to prevent the group from wandering too far. We would also like to thank SMSLib for releasing such a useful toolkit which we were able to build our server portion upon. Finally, we are very thankful for the many participants of Google Groups Android Development, their help assisted us many times to answer questions which seemed to have no answer and no solution.

References

1. "FrontlineSMS." FrontlineSMS. kiwanja.net. 28 May 2008 <http://www.frontlinesms.com/>.
2. Kasvi, Keijo. "SMS Server Tools 3." SMS Server Tools 3. 11 May 2008. SMS Server Tools 3. 28 May 2008 <http://smstools3.kekekasvi.com/>.

3. Kovalenko, Anton. "SMS Reception Center." SMS Reception Center. 08 Feburary 2008. Sw4me. 28 May 2008 <http://sw4me.com/wiki/SMSReceptionCenter?v=vsd>.
4. "MSR India SMS Toolkit." MSR India SMS Toolkit. 2008. Microsoft. 28 May 2008 <http://www.codeplex.com/smstoolkit>.
6. Lesh, Neal. QuickSurv. Personal Contact. <neal@equalarea.com>.
7. Pacqué, Michel. "Malaria: Prompt Treatment Saves Lives." Maximizing Access and Quality (MAQ) Initiative. 09 August 2005. USAID. 11 Jun 2008 <http://www.maqweb.org/techbriefs/tb19maltreat.shtml>.
8. Reardon , Marguerite. " Emerging markets fuel cell phone growth." CNet News. 14 February 2007. CNET. 28 May 2008 <http://news.cnet.com/Emerging-markets-fuel-cell-phone-growth/2100-1039_3-6159491.html>.
9. "SMSLib." SMSLib. 18 May 2008. SMSLib. 28 May 2008 <http://smslib.org/>.
10. Tseng, Chwan-Lu, Joe-Air Jiang, Ren-Guey Lee, Fu-Ming Lu, Cheng-Shiou Ouyang, Yih-Shaing Chen and Chih-Hsiang Chang. "Feasibility Study on Application of GSM-SMS Technology to Field Data Acquisition." Computers and Electronics in Agriculture 53.1 (August 2006): 45-59.
11. "User Manual." GSM-CONTROL SMS Gateway Software. September 2007. KLINKMANN AUTOMATION. 28 May 2008 <www.klinkmann.com>.
12. Veeraraghavan, Rajesh. "Warana Unwired." Warana Unwired. Microsoft Research. 11 Jun 2008 <http://research.microsoft.com/~rajeshv/warana.htm>.