I.J. Information Technology and Computer Science, **2013, 05, 22-28**
Published Online April 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2013.05.03

iText – SMS-based Web Services for Low-end Cell Phones

Juan Li, Justin Anderson, Matti Kariluoma, Kailash Joshi, Prateek Rajan, Pubudu Wijeyaratne
North Dakota State University, Fargo, USA
E-mail: (j.li, justin.anderson.2, matti.m.kariluoma, kailash.joshi, prateek.rajan Pubudu.Wijeyaratne)@ndsu.edu

***Abstract*—** With the development of mobile wireless technology, more people are using their smart phones to browse Web sites, do social networking, and play online games. However, a significant fraction of people in developing regions are still using simple low-end devices with limited processing and communication capabilities. Access to Internet contents and services is currently available only for Internet-enabled, Internet-capable smart devices. To address this problem, in this paper, we propose a Short Messaging Service (SMS)-based service to enable low-end cell phones to access Internet and Web services. In particular, we designed a service called iTex, allowing users to use Google web services through text messages even if they do not own Internet-enabled and Internet-capable smart devices.

***Index Terms*—** Short Messaging Service, Email, Mobile Computing

I. Introduction

As time passes, technology is growing and moving faster. The most important and common part of technology in our life is mobile-phone technology. We take a mobile phone with us in everywhere that we go and use it on a daily basis. It is part and parcel of our daily life. Mobile phones have been around for quite some time, but as time go on, mobile phones continuously gain many features. Mobile phones started as simple devices that had only numbers, and most people used them for emergencies only. Now, cell phones have many additional features, such as portable media players, low-end compact digital cameras, pocket video cameras, and GPS navigation units to form one multi-use device. Modern smart phones also include high-resolution touch screens and web browsers that display standard web pages as well as mobile-optimized sites. High-speed data access is provided by Wi-Fi and Mobile Broadband. The mobile phone is an exceptionally useful tool that advances personal communication beyond all expectations.

Despite the fact that smart phones are gaining increasing power and popularity, a significant fraction of mobile devices in developing regions are still simple low-cost devices with limited processing and communication capabilities [1]. This is especially true for developing countries. In fact, according to Pew Internet Project [2], only one third of American adults own smartphones, among the 83% of US adults who have a cell phone of some kind. Due to a combination of social and economic factors, voice and SMS will likely continue to remain the primary communication channels available for a non-trivial fraction of the population in developing regions [1].

Nowadays access to Internet is limited to PC users and smart phone users, which forms about 5% of our population [3]. People at the bottom of the pyramid do not have access this huge source of information and services that can dramatically improve their lifestyle [4]. To break this barrier and enable even non-Internet capable devices to get the value of the Web, we propose a SMS-based service that enables users to access Web services though short messages. Our SMS service bridges the gap between smart phone and low-end cell phones that do not have Internet capabilities. Our solution enables users with low-featured mobile devices to access to Web content anywhere anytime. This has a good economic impact too, because a smart phone is at least 10 times expensive than a low end phone. Moreover, data plan of Internet access is pretty high. Our solution enables users to be online with minimum cost. Of the available web services, we chose to implement access to Google's services for our project. This does not prevent one from extending our methods to other services, such as Yahoo, eBay, Amazon, etc.

The remainder of this paper is organized as follows: In Section 2 we detail the background knowledge and related work. Section 3 describes the design of the proposed system, iText. In Section 4, we evaluate the proposed methods and show their effectiveness with survey-based usability study. Concluding remarks are provided in Section 5.

II. Background and Related Work

2.1 SMS Communication

Short messaging service (SMS) is a mechanism of delivering short messages over mobile networks. It is a store-and-forward way of transmitting messages to and from mobile devices. The message (text only) from the

sending mobile device is stored in a central short message center (SMC) which then forwards it to the destination mobile phone. This means that, when the recipient is not available, the short message is stored and sent later. Each short message can be no longer than 160 characters. These characters can be text (alphanumeric) or binary Non-Text Short messages [5].

SMS technology is supported by 100% of GSM handsets and is, thus, available for anyone who has a GSM handset. SMS was originally developed for person-to-person messaging; however, it has been widely deployed outside this scope. It is now being used to send SMS notifications for new voice mail, email, and fax messages, allowing remote monitoring of services where an SMS notification is sent out notifying the administrator that a server is running out of resources or that a fault has been detected.

Rapid growth of services and systems that are built around SMS has led to the development of SMS APIs which help developers provide their service to intended users in an easy way.

2.2 Related Work

Although SMS text messaging is the most widely used data application in the world, with 3.6 billion active users, or 78% of all mobile phone subscribers [6]. SMS approaches to providing internet services to low-end mobile phones are not still very common. In this section, we list some existing SMS services.

Currently, there are email service providers providing "Push"-based email notification through SMS. For example, Gmail allows configuration of the inbox to send an SMS alert to a phone whenever users receive a new email. This *push*-based notification needs the email provider's support, and it may frequently send emails which bother the user.

Google SMS applications [7] and Yahoo Search for Mobile [8] allow users to send search queries by SMS and results are sent back via SMS again. Due to the limitations of the length of SMS, the usefulness of the application is in question. A recent research uses machine learning approaches to return users with summarized results [1].

Most of the companies develop separate mobile applications that integrate with their system to provide SMS based services. For example, many banks provide SMS service to their customers. Wells Fargo, Bank of America, U.S. Bank, and Citi Bank are some of the banks that use SMS banking to cater services to their customers. Banking services are operated using both push and pull messages. Push messages are those that the bank chooses to send to a customer's mobile phone without the customer initiating a request for the information. Typically, push messages could be either mobile marketing messages or messages alerting an event for the customer's bank account, such as a large withdrawal of funds from the ATM, a large payment using the customer's credit card, etc.

Pull messages are initiated by the customer, using a mobile phone, to obtain information or perform a transaction in the bank account. Examples of pull messages for information includes an account balance inquiry or requests for current information (i.e., currency exchange rates and deposit interest rates) published and updated by the bank [9].

III. System Design

Fig. 1 illustrates the architecture of the iText service. A detailed description about these components is presented in this section. Any mobile phone with a text-messaging capability can interact with the iText service. All text messages should be sent to a particular phone number. All incoming text messages are first received by a SMS daemon program. The daemon forwards the incoming message to the call-back URL that was provided during the registration process. When a call has been made to the call-back URL, the code behind the Call Back URL saves the text message and its information to a SQL server database. The iText handler processes these requests according to the order they are received. Once a message has been processed, the iText handler sends the reply message as an http request. The user's phone number and the message to be sent are added to the header of HTTP request. As the final step, the server delivers the message to the user's mobile phone.

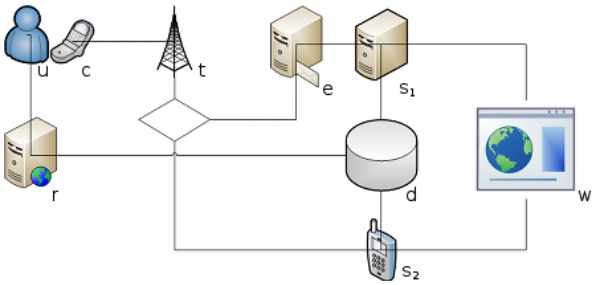

Fig. 1: iText system diagram

Legend: (u) User. (c) Cellphone. (t) Cellphone Tower. (e) Email Server. (s1) Application Server. (s2) Smartphone. (r) Registration Server. (d) Centralized Database. (w)Web Services.

3.1 SMS Server

We have investigated many potential ways to implement a server to receive and respond SMS messages. We found the following to be most promising:

1. SMS capable phone with the server application

2. SMS phone with serial connection to computer with server application

3. Application server which listens on an Email server, SMS are sent using an SMS to Email gateway (a service that translates the protocols)

4. Add-on card for computer which participates with the telephone network

5. Application server which listens on a web server providing one of the above

From these five candidate approaches, we adopted the first and the third approach, because the latter two require a paid account and the second was overly complicated, especially given that we had access to a smart phone to run our application server. The second method is more appropriate for an implementation with a more basic, low-end cell phone as the SMS gateway.

iText Daemon

A third-party program, "links"[10] was used to render HTML pages to plaintext. The links binary was not available for the phone gateway; a resource server was created to render the HTML remotely, where the "links" program was available. This resource server runs on the same machine as the email gateway.

Phone Gateway

We wrote our iText phone daemon for a Nokia n900 smart phone running the Maemo [11] operating system, a derivative of Debian GNU/Linux. The authors limited the choice of programming language to those available from the Maemo distributions package manager: C, C++, bash, perl, and Python. Python was chosen for three reasons: portability of code, available libraries, and ease of data structure expression.

When the iText phone daemon receives an SMS message, it sleeps for 1 second (otherwise the phone's carrier, AT&T, will not deliver the message due to rate limiting), then processes and responds to the message. The messages are processed as they travel through the phone's subsystem [12], therefore if the iText daemon is not running when a message is received, it will not be processed.

Email Gateway

A separate machine was used to host both an Email Server and the iText email daemon. The iText email daemon periodically checks its assigned mailbox and processes each unread email, marking it as read after processing.

3.2 User Authentication

We investigated three methods of accessing a user's credentials for queries to restricted web services:

- User sends username & password with SMS request
- User stores username & password in a centralized database
- User stores an authentication token in a centralized database

An SQL database was set up on a centralized server where the registration servers could store a user's identification data. The iText daemons then accessed this centralized database, using the reply-to address as an index into the SQL database for the credentials to access email and calendar services on behalf of the user.

3.2.1 Authentication with Username & Password

Google Data API

The Google Data API is based upon the Google Data protocol. The Google Data protocol extends the Atom 1.0 [13] & RSS 2.0 [14] syndication formats, and the Atom Publishing Protocol (APP). [15]

The Google Data protocol extends these standards in various ways, using extension mechanisms built into the standards. Feeds conform to either the Atom or RSS syndication formats. The publishing model conforms to the Atom Publishing Protocol.

Accessing Gmail Data

Gmail's Atom RSS Feed https://mail.google.com/mail/feed/atom/ is accessed using HTTP GET parameters. When requesting a feed the default data format is XML. One can specify an alternative format using the HTTP GET parameter .alt to receive data in the RSS *.alt=rss* or JSON *.alt=json* format.

In order to receive the data, a Base64 encoded "Username" and "Password" header must be appended to the GET request.

3.2.2 Authentication with Tokens

Google uses a technology known as Open Authentication (OAuth) for sharing information between a user and an application. ‘OAuth is an open standard for authorization. It allows users to share their private resources (e.g. photos, videos, contact lists) stored on one site with another site without having to hand out their credentials, typically supplying username and password tokens instead. Each token grants access to a specific site (e.g., a video editing site) for specific resources (e.g., just videos from a specific album) and for a defined duration (e.g., the next 2 hours). This allows a user to grant a third party site access to their information stored with another service provider, without sharing their access permissions or the full extent of their data.” [16, 17, 18]

Google's email service, Gmail, does not yet fully support access via OAuth. A user's unread email can be accessed through a deprecated version of OAuth (version 1), but until OAuth (version 2) is implemented,

access through the Atom RSS feed is recommended [19].

Open Authentication

OAuth uses tokens. A token is a string of characters that:

- Gives permission to an application to make API calls to retrieve and manipulate data
- Determines what information can be accessed

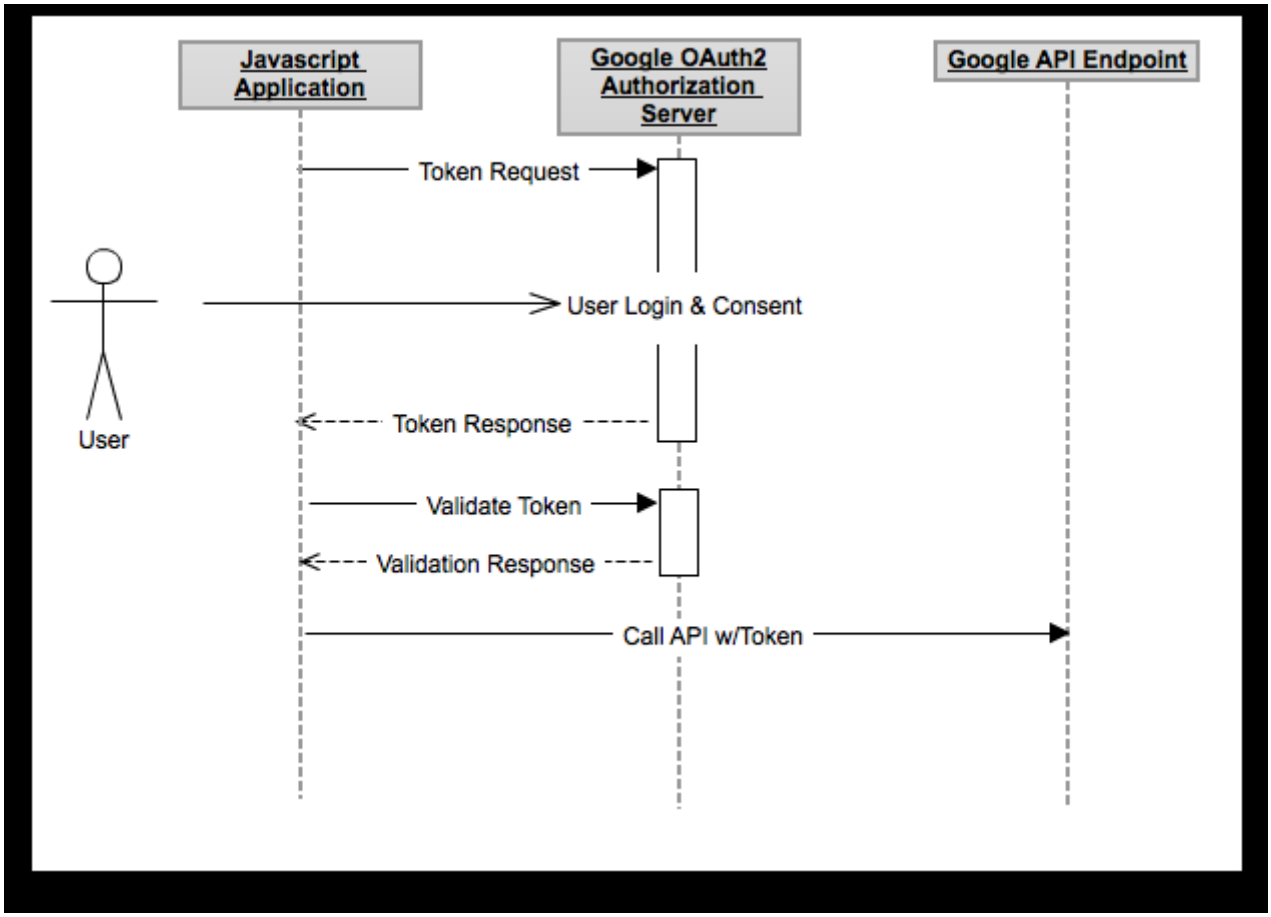

Fig. 2: Flow chart depicting retrieval of an OAuth token [17]

In Fig. 2 we can see the process of acquiring a token:

1. The user clicks a link or the application redirects the user to a log in page on Google's server.
2. The user puts in his/her credentials and clicks a button to agree to let the application manipulate their data.
3. Google redirects the user back to the application with the granted token.
4. The application then stores this token for further use.
5. Finally, the application can make various API calls using the token.

Google uses a data format called *Javascript Object Notation* (JSON) to store the token and other related information.

Accessing Google Calendar Data

In our project, we used Google's OAuth to access a user's Google Calendar information. We wrote a user creation webpage in PHP that instructs the user follow a link to a login page, on Google's server, which return the user to our page with a token. We then store this access token in our centralized database.

Google's OAuth implementation requires a named domain for the return page. One of our user creation pages was written in C# in order to test the validity of our methods across platforms.

The C# implementation runs on a server separate from the server registered with Google's OAuth. In order to access and use the data being sent to the registered server (without writing temporary SQL rows), a JavaScript redirect script was written to access the token, after which it is written to the centralized database.

IV. Usability Evaluation

To evaluate the usability of the proposed service, we designed a survey study. Students from different college of NDSU were invited as volunteers to use and comment the system. In particular, we invited students from the College of Science and Mathematics, the College of Engineering and Architecture, College of Business, College of Agriculture, Food Systems, and Natural Resources, College of Arts, Humanities, and Social Sciences, and College of Pharmacy, Nursing, and Allied Sciences. Based on their self-evaluation, the participants' computing skills and familiarity with mobile wireless devices were categorized as high computing skills, average computing skills, and low computing skills. We chose participants from different groups and levels because it is important to get a balanced perspective as best can be achieved to fairly evaluate the usability of the system.

Table 1: Question list

#	Question	1	2	3	4	5
1.	Overall I find the system easy to use					
2.	I was able to complete the tasks easily using the system					
3.	Whenever I made a mistake, I find it easy to recover					
4.	The system provides sufficient error handling to help fix a problem					
5.	The system has all the features and functions I expect					
6.	The system response time is acceptable					
7.	The information provided by the system is easy to understand					
8.	The information provided by the system is helpful in completing tasks					
9.	SMS commands are easy to remember and verbiages make perfect sense					
10	I would happily use this system again					
11	I would recommend this system to the people I know					

30 volunteers participated in the survey. The volunteers were asked to complete a series of pre-defined tasks with no aid and then to fill out a short questionnaire which was designed using a Likert scale [20], giving the volunteers five options varying from strongly disagree to strongly agree. The list of pre-defined tasks was as follows:

1. Send a text to a particular number and follow the instructions (Type CODES to receive all the available SMS commands)
2. Add an email account following the ADD command
3. Check how many messages you have using the CHECK command
4. List all messages in your inbox with the LIST command
5. Read an email from your inbox by following the READ command
6. Send an email to someone using the SEND command
7. Remove the email account you added using the REM command

The Likert scale for the questionnaire is shown in Table 1, where *5* is strongly agree and *1* is strongly disagree.

Once all of the volunteers had completed the questionnaires, the marks for each question and user were averaged to determine the average grade for the system.

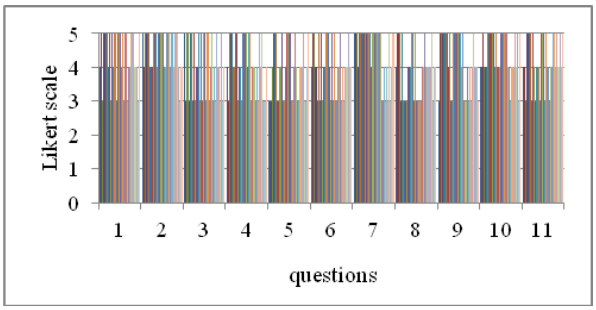

Fig. 3: Average score for each answer

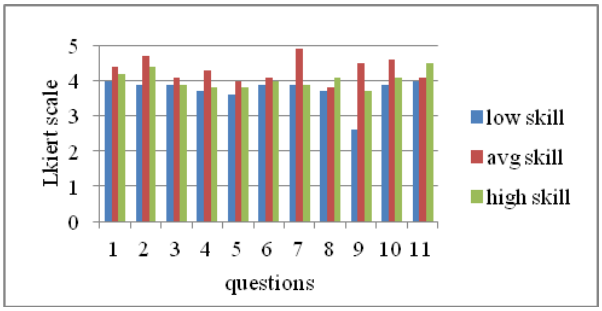

Fig. 4: Average score for each category

From the results gathered, although the sample size is small, the feedback gained about the usability of the system was positive with volunteers consistently rating

most questions as agree or strongly agree. The graph in Fig. 3 shows that the average score for each answer was around four with some answers being close to five. This suggests that the usability of the system is good with slight room for improvement. The results also show that the system conveys information to the user effectively and that users can become productive with the system in a reasonable amount of time.

From the chart shown in Fig. 4, it is clear that the different groups found the usability of the system different with the low-computing skills group finding it the least satisfactory. It should be noted that, despite the low-computing skills group finding the usability not as good as the other groups, the mean answer is about 3.8, which is still an encouraging outcome for the evaluation. A possible way to improve this score is to conduct an exercise where feedback can be gathered from the low-computing skills group about how to improve usability and to implement the necessary changes to the system in order to achieve a higher level of usability. By talking with the volunteers after they completed the questionnaire, one problem seemed to arise with the system usability; i.e., volunteers suggested that, rather than showing just the subject of the message, it would be nice to show the entire message body. Another suggestion was to make it possible to read attachments, such as MS Word files, and deliver their content via a text message.

V. Conclusion

As mobile phone ownership continues to increase rapidly in many countries worldwide, they have become an integral part of the modern world, providing human connectivity in a way never before possible. Besides their original intent—making telephone calls wirelessly, nowadays, mobile phones are also loaded with lots of other features. However, despite the increasing power of smart phones, a significant fraction of mobile devices in developing regions are still simple low-cost devices with limited processing and communication capabilities. The aim of this project is to provide a solution that allows users to access Internet services via SMS text messages from a non-internet capable mobile phone. This was achieved with a SMS-based system, iText that was built and tested. The evaluation proved that the project was a success while leaving the potential for the system to be developed further.

References

[1] J. Chen, L. Subramanian, and E. Brewer, “SMS-Based Web Search for Low-end Mobile Devices,” in MobiCom ’10. New York, NY, USA: ACM, 2010, pp. 125–136.

[2] Pew Research Center's Internet & American Life Project: http://www.pewinternet.org/.

[3] Traffic and Market Data Report - Ericsson, June 2012, http://www.ericsson.com/res/docs/2012/traffic_and_market_report_june_2012.pdf

[4] Geetha Manjunath, M M Revathi, "Rural Healthcare on lowend phones", May 6, 2011 – Tech Report: HPL-2011-63.

[5] Gupta, P. (2010). The Short Message Service: What, How and Where. Retrieved January 20, 2012, from Wireless Dev Net: http://www.wirelessdevnet.com/channels/sms/features/sms.html

[6] Tomi T. Ahonen, Alan Moore., "http://communities-dominate.blogs.com/brands/2011/01/time-to-confirm-some-mobile-user-numbers-sms-mms-mobile-internet-m-news.html".

[7] Google SMS applications: http://www.google.com/mobile/sms/

[8] Yahoo Search for Mobile: http://mobile.yahoo.com/search

[9] SMS Banking. (2011, December 16). Retrieved January 23, 2012, from Wikipedia: http://en.wikipedia.org/wiki/SMS_banking

[10] Twibright Labs. Links: Web browser, 2012. URL: http://links.twibright.com/.

[11] Nokia. Repository, 2012. URL: http://wiki.maemo.org/Repository.

[12] Cue. Sms from cli (command line)?, 2010. URL: http://talk.maemo.org/showpost.php?p=558430&postcount=57.

[13] R. Sayre M. Nottingham. The atom syndication format. Internet Engineering Task Force, 2005. URL: http://www.ietf.org/rfc/rfc4287.

[14] RSS Advisory Board. Rss 2.0 specification, 2002. URL: http://www.rssboard.org/rss-specification.

[15] B. deHora J. Gregorio. The atom publishing protocol. Internet Engineering Task Force, 2007. URL: http://www.ietf.org/rfc/rfc5023.txt.

[16] OAuth wikipedia: http://en.wikipedia.org/wiki/OAuth

[17] Google. Using oauth 2.0 to access google apis, 2012. URL: https://developers.google.com/accounts/docs/OAuth2.

[18] Eran Hammer. Introducion oauth2.0, 2010.URL: http://hueniverse.com/2010/05/introducing-oauth-2-0/.

[19] Google. Google data api faq, 2012a. URL: https://developers.google.com/gdata/faq.

[20] Likert_scale : http://en.wikipedia.org/wiki/Likert_scale

Authors' Profiles

Juan Li: Assistant Professor in Computer Science Department of North Dakota State University.

Justin Anderson: Undergraduate Student in Computer Science Department of North Dakota State University.

Matti Kariluoma: Graduate Student in Computer Science Department of North Dakota State University.

Kailash Joshi: Graduate Student in Computer Science Department of North Dakota State University.

Prateek Rajan: Graduate Student in Computer Science Department of North Dakota State University.

Pubudu Wijeyaratne: Graduate Student in Computer Science Department of North Dakota State University.

