ELMR: Efficient Lightweight Mobile Records

Arvind Kumar, Jay Chen, Michael Paik, Lakshminarayanan Subramanian
Dept. of Computer Science, New York University
arvind.kumar@cs.nyu.edu, jchen@cs.nyu.edu, mpaik@cs.nyu.edu, lakshmi@cs.nyu.edu

ABSTRACT

In this paper we describe Efficient Lightweight Mobile Records (ELMR), a system that provides a practical protocol for accessing and updating database records remotely from low-end mobile devices using the 140-byte SMS channel.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed Applications

General Terms

Algorithms, Design, Experimentation

Keywords

Cell Phones, User Interface, Healthcare, Compression

1. INTRODUCTION

In rural regions around the world, especially underdeveloped areas, it is often difficult to gain access to basic healthcare. In these areas much of the burden of healthcare delivery falls on community health workers (CHWs). The massive penetration of cellular services in these regions position mobile devices and applications to revolutionize the way healthcare is delivered by providing a channel for these CHWs to gain on-demand access to relevant data to improve quality of treatment. Most countries in Africa have over 50% cellular coverage [2] and a significant fraction of their rural population owns or has access to a mobile phone [6].

Several recent research and developmental efforts such as OpenRosa [5], OpenMRS [4], and Voxiva [7] have explored the use of mobile phones as a low-cost computing platform for distributed healthcare applications. However, these existing systems are not scalable and sustainable in developing contexts because the software implementation itself (reliant on TCP/IP and SQL) is typically too heavyweight for low-end mobile phones, calling for the additional expense of smartphones. They also depend on GPRS network connectivity which is feasible only in urban settings; in most rural settings only voice and Short Messaging Service (SMS) (140 byte packets) services are available.

Alternative efforts based around SMS exist. FrontlineSMS [1] provides SMS broadcast capabilities, and SMS-NIC [3] focuses on creating a reliable low bandwidth communications channel using SMS. However, neither of these is perfectly suitable for a mobile record application such as ELMR since they either provide the wrong abstraction and/or are not optimized for our main application requirement: cost.

To address these limitations we designed and implemented ELMR, a generic record system that is both feature-complete and cost effective. ELMR is SMS-based and runs on low-end mobile phones enabling health workers and patients in rural areas to fetch and update their patient records and doctors to remotely track their patients' health.

2. ELMR DESIGN

At a high level, the architecture of ELMR is simple: a client running on the user's mobile phone accesses the remote database by sending and receiving SMS messages to and from the server. The novelty of ELMR lies in the combination of its message encoding scheme, transmission protocol, and reliability/consistency management.

2.1 Restricted Set of Operations

We support only the following operations: `create`, `append`, `update`, `destroy`, `search`, `fetch`, and `aggregate fetch`.

A given command message contains the operation ID, the schema ID, bits reserved for control information and the payload containing the variable-length data sent to perform the operation specified by the operation ID.

2.2 Semantic Compression

ELMR employs semantic compression which reduces the number of messages exchanged between the client and server to complete an operation by reducing payload space required for each command's requisite data.

To better utilize the space in each SMS message we limit the types of fields in a schema. All fields in the form are represented by one of these data types: 1. *Date* 2. *Integer* 3. *String* 4. *Boolean* 5. *Multiple Choice*

We optimize by automatically assigning the minimum number of bits to each datatype (e.g. 1 bit to booleans) during schema encoding and decoding. In addition, each integer and date field carries a precision modifier and each multiple choice contains a finite number of options, which aids in correctly assigning the minimum number of bits to make the most of the 140-byte SMS payload. Variable-length strings have a 1-byte length value prepended to them.

2.3 Lightweight SMS Reliability

We propose a thin layer of best-effort reliability that does not guarantee 100% reliability. Our protocol is essentially stop-and-wait per *session*, and outlined below:

Session Definition:
1. Every session has a unique 8 bit identity.
2. A session contains at most 16 messages.
3. Each message per session contains a 4-bit sequence #.
4. A session has 3 phases.

Phase 1: The client initiates a session and then sends at most 16 messages to the server. The first of these messages has a single bit set indicating that it is the first message in the session and uses its sequence number field to indicate the total number of messages in the session. The server waits for SMS messages and does not send any acknowledgements.

Phase 2: After either a certain amount of time passes and the session times out or all messages are successfully received by the server, the server sends a message with a 16-bit ACK vector. The vector contains 1's for the sequence numbers which have been received, and 0's for those wihch have not. The client retransmits these missing messages, if any, when the ACK vector is received. If the server is waiting for missing messages but does not receive them, it attempts one more round of sending an ACK vector and waiting for a response, then fails the session. Once the server has received all messages, either in the first round or in a subsequent retransmission round, the ACK may either be sent immediately or piggybacked on a response in phase 3. If messages are still not received the server resends a NACK one last time. If messages are received the server sends an ACK.

Phase 3: The server sends up to 16 messages in response to the client in a session. In this case, the client does not ACK the server messages unless the client receives some of the messages and others are lost. This is the mirror of phase 2 from the client side. If all messages are received or all messages are lost, the client does not send a NACK.

In this protocol, a session may not complete under three scenarios: a) All messages from client to server are lost. b) All messages from server to client are lost. c) Transmission of the initialization vector is lost in either direction. In any of these cases, the client times out and asks the user whether he wishes to retry the session.

2.4 User Driven Consistency

Since ELMR uses an append only database model we don't directly deal with consistency and choose to leverage user driven consistency where inconsistencies are either handled by the system administrators or the users themselves. In scenarios where explicit consistency guarantees are required, timestamps and conflict resolution mechanisms can easily be incorporated into our design.

3. EVALUATION

We performed a preliminary evaluation of ELMR using real medical forms used by hospitals in Africa. Our evaluation is composed of two parts: we evaluate the effectiveness of ELMR at reducing the size of messages and analyze the message overhead of our reliability protocol.

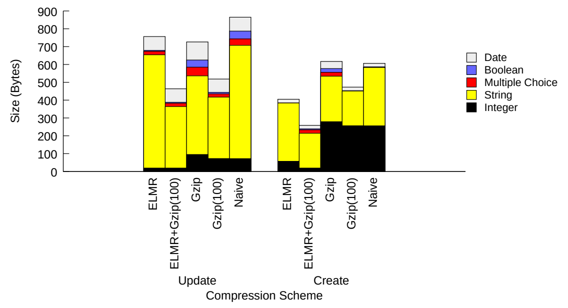

Figure 1: Field sizes after various compression schemes

Figure 1 illustrates the difference in average size per form between naive un-optimized text forms, simple gzip compression, aggregate gzip compression, and ELMR. As the effectiveness of semantic compression is a function of the data types, the results are broken down according to the types of the original forms. Gzip is the average compression rate for single messages. Gzip(100) refers to the average operation across 100 aggregate operations. ELMR is the benchmark where only semantic compression is used. We observe that semantic compression improves upon the uncompressed version for both Intake and HIV/TB. We can see that the uncompressed performs the most poorly in both cases. The Gzip compression performs only slightly better than uncompressed for HIV/TB and worse for Intake, and the Gzip(100) further compacts strings across messages. ELMR is not currently optimizing string fields using semantic compression, but even if we simply compress the strings in ELMR using Gzip we achieve improved results as seen in ELMR+Gzip(100) reducing the original payload size by nearly 50% in the aggregate case.

For the evaluation of reliability and consistency protocol we simulated our protocol over a range of loss rates from 0 to 20%. We observe from the results that the overhead is low even for loss rates of up to 20%.

4. REFERENCES

[1] FrontlineSMS. http://www.frontlinesms.com/.
[2] International Telecommunication Union. http://www.itu.int/.
[3] E. Oliver. Exploiting the Short Message Service as a Control Channel in Challenged Network Environments. In *Proceedings of the third ACM workshop on Challenged networks*, 2008.
[4] OpenMRS. http://openmrs.org/wiki/OpenMRS.
[5] OpenRosa. http://www.openrosa.org/.
[6] M. Paik et al. The Case for SmartTrack. *IEEE/ACM Conference on Information and Communication Technologies and Development (ICTD)*, 2009.
[7] Voxiva. http://www.voxiva.com/platform.php.