

# Characterizing the Transport Behaviour of the Short Message Service

Earl Oliver

David R. Cheriton School of Computer Science  
University of Waterloo  
Waterloo, Ontario, Canada  
eaoliver@uwaterloo.ca

## ABSTRACT

We build an efficient and reliable data transport protocol on top of the Short Message Service (SMS). We conduct a series of experiments to characterize SMS behaviour under bursty, unconventional workloads. This study examines how variables such as the transmission order, delay between transmissions, the network interface used, and the time-of-day affect the service. We present the design and implementation of our transport protocol. We show that by adapting to the unique channel conditions of SMS we can reduce message overheads by as much as 50% and increase data throughput by as much as 545% over the approach used by existing applications. Although the transport protocol can provide efficient, low-bandwidth, moderate-latency data communication between mobile devices throughout the world, we believe that this work has the largest potential impact in developing regions where SMS is often the only option for wireless data communication.

## Categories and Subject Descriptors

C.2.2 [Network Protocols]

## General Terms

Design, Experimentation, Reliability

## Keywords

short message service, transport protocol, control channel

## 1. INTRODUCTION

The Short Message Service (SMS) is a phenomenally popular global wireless service. In 2008 alone, nearly 3.5 trillion SMS messages were sent [29]. In the United States, SMS usage tripled between 2007 and 2009, with over 1 trillion messages being sent in the US in 2008 [5]. Today SMS has grown beyond its traditional use as a mobile-to-mobile text messaging service and has become an integral component of

many mobile applications. For example, the service is commonly used to conduct electronic surveys, provide e-voting services, send calendar notification, search the Internet, and exchange status updates with servers on the Internet. The list of uses continue to grow with no sign of slowing down.

We build an efficient and reliable data transport protocol on top of SMS. This work builds significantly over preliminary work [26] both in the depth of analysis and completeness and evaluation of the protocol design. Specifically, this paper makes two major contributions.

First, we motivate the design of our transport protocol through a characterization of the SMS service of a major Canadian cellular network provider. This paper explores the complex behaviour of this service from the perspective of mobile applications that use SMS to exchange data between devices. Through experimentation with a variety of commodity smartphones and USB tethered cell phones, we classify the channel characteristics of SMS and identify key variables that affect its performance. Unlike networks such as the Internet, SMS can introduce significant delays, messages are rarely lost, message reordering is common, and the service does not suffer from congestion drops. Thus conventional transport protocols, such as TCP, are unsuitable for use over SMS. By examining the service from the perspective of a mobile device sending bursts of messages, this paper differentiates itself from previous work that has focused purely on aggregate statistics as observed by service providers. Our measurement tools are available for further study of other cellular networks [8].

Second, we design and implement an efficient SMS-based data transport protocol that we call *SMS-TP*. SMS-TP provides reliable data transport while attempting to minimize message overhead and maximize data throughput of successful data transfers. We have implemented SMS-TP as a stand-alone library in Java Micro Edition. Our implementation is compliant with the Java Connected Limited Device Configuration (CLDC) and can therefore be embedded into a wide range of applications running on Java enabled cell phones, smartphones, and PC environments (using a USB tethered cell phone). The current implementation can be downloaded for free [8]. While we do not claim that our transport protocol is optimal, our protocol significantly outperforms existing, available, alternatives. We show that by adapting to the unique channel conditions of SMS, SMS-TP can reduce message overhead by as much as 50% and increase data throughput by as much as 545% over the existing method used by existing mobile applications to exchange large amounts of data over SMS.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MobiSys'10, June 15–18, 2010, San Francisco, California, USA.  
Copyright 2010 ACM 978-1-60558-985-5/10/06 ...\$10.00.

This paper begins with an overview of SMS and prior work. Section 3 describes our measurement study and we present out experimental results in Section 4. Section 5 describes the design of SMS-TP. We evaluate the protocol in Section 6.

## 2. MOTIVATION

### 2.1 The benefits of SMS

The growing adoption of SMS in mobile systems is due to a number of factors that make SMS a desirable medium for mobile data exchange:

- **Ubiquity:** SMS is a globally accepted standard and available to cellular subscribers in nearly every developed and developing country in the world. Today, GSM service alone is available in over 220 countries across 860 networks [12]. Moreover, alternatives to SMS, such as the Enhanced Message Service (EMS) and GPRS/EDGE, are poorly supported or sparsely deployed outside of developed regions [18].
- **Reliability:** Like traditional mail servers, SMS uses a central server called the *Short Message Service Center* (service center) to provide reliable store-and-forward message delivery between two mobile devices [19, 1]. When a message is accepted for delivery, the sender is reasonably guaranteed that the message will either reach its destination or expire after a few days.
- **APIs:** The ability to programmatically send and receive SMS messages is a standard feature of all major mobile development platforms [24]. SMS Gateways have also proliferated, thus enabling applications on the Internet to seamlessly communicate with mobile devices.
- **Device-to-device communication:** SMS provides mobile devices the ability to communicate with each other directly. Unlike cellular data services, where devices are typically behind a NAT and allocated an IP address using DHCP, SMS provides mobile devices the ability to communicate with each other directly using their phone number. Thus mobile devices can both send and receive data without the need to poll a central server for updates.

Although many mobile applications are adopting SMS to exchange data, we observe one distinct commonality: nearly all applications constrain their use of SMS to its 140 byte payload (160 7-bit encoded characters). Several recent examples include [3, 21, 34, 37]. Applications that must exchange more than 140 bytes of data do so by fragmenting their data and sending it using a series of messages [35, 33]. These applications use a simple stop-and-wait protocol, which we describe in detail in Section 6.1.1. As mobile devices continue to become an integral part of daily life [16], we believe that applications will increasingly exploit the benefits of SMS and exchange increasingly larger amounts of data - particularly in developing regions where the cellular network is often the only means of communication. Our work aims to make efficient use of this communication channel.

### 2.2 Alternatives to SMS

Today, there are two major alternatives to SMS that mobile applications can use to reliably exchange data: cellular data services and the EMS. As a data channel, SMS is greatly inferior to EVDO, GPRS/EDGE, and other cellular data services. SMS has significantly lower data rates, high latency, a small fixed message size, and messages can be lost during transport. However, data services are expensive and sparsely deployed in many parts of the world. In developing regions, cellular data services are often non-existent [18], leaving SMS the only cost-effective, viable option for data communication.

The EMS is an application level extension to SMS [2]. Although EMS is defined to support up to 255 concatenated messages, the limited set of devices that support EMS typically constrain EMS messages to 918 bytes. EMS also suffers from poor service provider interoperability, and EMS messages typically cannot be exchanged between devices on different cellular networks. While EMS is an improvement over the existing approach and results in a lower message overhead and higher throughput, the lack of standardization and limited data size make it unsuitable for use as a general-purpose transport layer.

### 2.3 Prior work

Previous studies on SMS have primarily focused on the security and vulnerability of the service. The feasibility of denial-of-service attacks and bulk messaging are examined in [6, 36, 20]. Further work [23, 13, 10, 14] examines the increased load on the cellular network and the GSM signaling channel introduced by SMS.

In [39, 22] Zerfos et al. examine the characteristics of SMS from the perspective of a cellular service provider. In this work, Zerfos presents an analysis of SMS traces collected by a service provider in India over a three week period. The network traces consist of over 59 million messages exchanged by more than ten million users, which at the time represented approximately 10% of India's total mobile subscribers. In this analysis, Zerfos classifies the current uses of SMS and measures how conversation threads progress across a series of messages. This work provides a preliminary classification of the behaviour of messages as they traverse the cellular network. In particular, the authors observe that nearly 5.1% of messages are lost during transit due to expiration or denial of delivery. They found that 73.2% of messages reach their recipient within a ten second delay, 17% require more than one minute, and the remainder take over an hour and a half. Their study also examines the service under flash crowd scenarios as regularly experienced by cellular networks during major holidays.

While this work provides valuable insight into the aggregate behaviour of SMS, it lacks the critical details needed to design a transport protocol. We therefore conduct a measurement study to acquire the necessary data to design a transport protocol. Our work complements the Zerfos study by examining SMS under bursty workloads and by examining the root causes of variation.

## 3. CHARACTERIZATION

The goal of this section is to define the channel characteristics of SMS. From this characterization we can derive algorithms that make efficient use of the channel. We begin this section with a system overview that motivates the key

characteristics of our study, followed by a discussion of our experiment parameters and methodology.

### 3.1 SMS overview

We begin our characterization of SMS with a brief overview of the service and describe the process of sending a point-to-point SMS message from one mobile device to another. This discussion is meant to highlight the variability inherent to the cellular network and motivate our experimental approach. Although this discussion is made in the context of GSM, SMS operates in a similar fashion over CDMA-based networks [31].

Consider a mobile device, otherwise known as a *mobile station* (MS), in an idle state in a particular cell. To send an SMS message, the MS begins by initiating a channel request over the random access channel of the *base transceiver station* (base station) that it is currently associated with. Since the random access channel is a shared channel, communicating over it can introduce random back-off delays due to the use of slotted ALOHA. The *base station controller* responds to the request by allocating a dedicated channel, allowing the MS to initiate transmission of its message. The transmitted message is relayed through the base station and base station control to the *mobile switching center*. The mobile switching center then forwards the message to the service center where the message is stored waiting for transmission to the recipient. Further details can be found in [32, 28, 7].

After sending the message, the MS returns to an idle state. While idle and *camped* in a cell, the MS monitors both the paging channel and the broadcast channels for changes to the settings of the current cell and the neighbouring cells. If a neighbouring cell has been identified as optimal, the MS will associate with the new cell’s base station. The details of base station selection are further discussed in [32].

To deliver an SMS message to the final recipient,  $MS_{dest}$ , the service center signals the mobile switching center that it has a message to deliver to a specified MS. The mobile switching center then requests base station controllers in the location area to page the  $MS_{dest}$ . Once the  $MS_{dest}$  receives the page, it requests a dedicated channel over the random access channel as before. Once the  $MS_{dest}$  receives a dedicated channel it replies to the mobile switching center with a page response. The mobile switching center then begins to forward the SMS message to the  $MS_{dest}$ .

### 3.2 Channel characteristics

Our study focuses on understanding the following characteristics of SMS from the perspective of mobile applications using the service: transmission time, delay, loss rate, and message reordering.

The transmission of an SMS message requires that the MS first allocate a dedicated channel for transmission. Provided that the cellular network is provisioned to handle the channel request, transmission time is primarily affected by the cellular signal strength of the MS and delays introduced by random back-off when requesting a channel over the random access channel [28, 7]. The transmission time of a message is measured as the time required for an application to make a blocking call into the OS, for the device to establish a dedicated channel, and to successfully transmit the message. Occasionally a message may fail to transmit due to channel contention on the random access channel or failure to allocate a dedicated channel. Our study also examines the

transmission failure rate and the time required for a transmission failure to be detected.

Every SMS message must enter the service center before it is transmitted to the  $MS_{dest}$ . The message is delayed within the service center before the  $MS_{dest}$  responds to a broadcast page, establishes a dedicated channel, and responds to the page. If a message cannot be delivered to the receiver, further delays may be introduced while the service center waits to reattempt delivery. We measure the delay of an SMS message as the time between the sender initiating transmission and the receiver receiving the message.

Like other store-and-forward protocols, the service center is responsible for storing a message until it can be successfully delivered to its recipient(s). The message is not deleted from the service center until it is successfully delivered or cannot be delivered and expires. The expiration time of an SMS message is controlled by the cellular provider and is typically set to one day. Although  $MS_{dest}$  can artificially inflate the loss rate by disabling the cellular radio or by leaving coverage, we assume that MSs remain within cellular coverage and are not engaged in a call or other activity that would inhibit the ability to establish a dedicated channel and exchange SMS messages. We measure the loss rate as the probability that a message is accepted by the network but not delivered to the  $MS_{dest}$ .

Messages may be reordered within the cellular network for a variety of reasons. Messages may traverse different paths that exhibit different transfer times, path lengths, and capacities. To increase capacity and robustness, the service center may be replicated; reordering may then occur as a natural side effect of parallelism. We consider two metrics when measuring reordering. The first metric considers the percentage of messages that are simply out-of-order. A message is out-of-order if its sequence number is less than the highest sequence number of its predecessors. For example, if sequentially numbered messages 0,1,3,2 are received, message 2 is reordered, which indicates a 25% message reordering rate. This is equivalent to the reordering definition in [27], where “late” packets are declared reordered. Our second reordering metric considers the overall queuing delay introduced as a result of reordering. Referring to our previous example, if message  $i$  was received at time  $t_i$  then the overall delay introduced by reordering would be  $t_2 - t_3$  since message 2 was clearly sitting in a transmit queue while 3 was delivered.

### 3.3 Methodology

We evaluate the channel characteristics through a series of experiments to isolate parameters that can influence SMS.

#### 3.3.1 Setup

Our experimental setup consists of a pair of Nokia 3390 cell phones tethered to Linux PCs and three pairs of different SMS capable smartphones: the BlackBerry Pearl 8220 (Pearl), BlackBerry 8820 (8820), and the BlackBerry Bold 9000 (Bold). The three BlackBerrys allow us to run our test driver as a native application on the device, and thus bypass any overhead associated with the serial connection between the PC and the tethered cell phone. Each BlackBerry also contains a different cellular radio, which allows us to assess the effect that different network interfaces may have on SMS service. The Pearl contains a M6000 radio from FreeScale Semiconductor with a Sky77526 power amplifier

from Skyworks. The 8820 also contains an MMM6000 radio and a Freescale PMM6037 power amplifier. The Bold contains an RPF 09040B radio from Marvell Technology that supports 3G [30]. We equip each device with a SIM card from a major Canadian GSM service provider (Rogers Wireless [15]). Although our study is based solely on measurements from one service provider, we observed similar characteristics during preliminary experimentation in Paris, France. We expect that the networks in developed regions will exhibit different properties [39]. We will revisit the properties of alternate networks through simulation later in this paper.

On each smartphone we use the cellular network to provide accurate clock synchronization. The Linux PCs synchronize with a local NTP server. On each PC and smartphone we install a software test driver that is responsible for either sending or receiving SMS messages. The sending application formats SMS messages and transmits them to the receiver. Each SMS message contains a unique timestamp that represents the start of the experimental trial and a monotonically increasing message ID. This information, along with the transmission start and end times, are recorded in a file each time a message is transmitted. Through a simple user interface, the sending application can be configured to perform a variety of tests, which we will describe in the next section. Finally, this application provides trial statistics to aide the experimenter and manages each SIM card’s SMS quota<sup>1</sup>. The receiving application is responsible for parsing the received messages and extracting the timestamp corresponding to the trial and the message ID. This application also records the data to a file for later analysis. On the PC, we communicate with the cell phone using an open source application called Gammu [9].

On the smartphones, both the sending and receiving applications record the cell ID and signal strength of the base station that the device is currently associated with. This data is sampled at a rate of two Hz and is written to a file on each device. This information was not available on the cell phone.

Throughout all experiments, each device was plugged in to an external power source. Preliminary experimentation found that removing external power had no effect on any output variables or signal strength. However, it is conceivable that future energy-aware cellular radios could exploit periods where external power is available to improve QoS by, for example, increasing their transmission power level. We note this attribute of our testbed for posterity.

### 3.3.2 Parameters

The characteristics of SMS may be influenced by the following tunable control parameters:

- **Intra-burst time:** To counter denial-of-service attacks and other suspicious behaviour, the mobile switching center could deny a MS’s request for a dedicated channel if it makes too many or too frequent requests. We define the intra-burst time for a message  $m_i$  as the amount of time that has passed since successfully transmitting message  $m_{i-1}$ .
- **Transmission index:** The mobile switching center or service center could also introduce delays depending

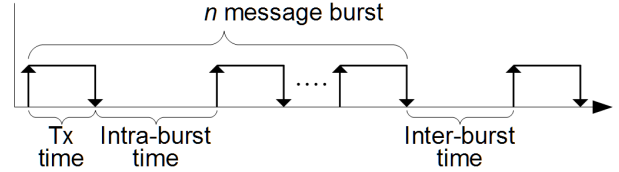


Figure 1: Relationship between intra-burst and inter-burst times.

on the number of messages recently or currently being serviced. We define the transmission index of a message as its order in a series of back-to-back messages, or *burst*. Bursts of messages are differentiated by an inter-burst time that is much larger than the intra-burst time. This relationship is illustrated in Figure 1.

- **Network interface:** Different cellular radios have different tolerance for low signal and interference. The operating system may also affect the device’s ability to respond to a page. We therefore believe that the choice of device could have an effect on the quality of service experienced when sending or receiving SMS messages.
- **Time-of-day:** The time of the day or the day of the week could influence random access channel contention, dedicated channel allocation, and workload on the service center; thus altering the channel characteristics.

In addition to these control parameters, the following parameters cannot be controlled.

- **Signal strength:** The signal strength between the MS and the base station can fluctuate for a variety of reasons, such as: movement by the MS, changing weather conditions, attenuation by buildings, or interference by other MSs. A drop in signal strength can cause the MS to associate with a new base station, which may have an impact on the service.
- **Base station:** Different base stations can exhibit different levels of random access channel contention and available capacity. Although base station selection heuristics are designed to select the “optimal” base station, sub-optimal selections can be made that impact SMS. The process of changing base stations may also affect SMS.
- **Local subscriber traffic:** Without access to service provider’s records<sup>2</sup> it is impossible for us to measure the effect that subscribers have on each other. We assume that the SMS infrastructure is sufficiently provisioned to ensure that subscribers have little affect on each other; however, at the edges of the network, channel contention can affect both the random access channel and allocation of dedicated channels. We compensate for local fluctuations by experimenting over long periods.

<sup>2</sup>Although some service providers have been willing to share data [39], none of the service providers we contacted were willing to share data pertaining to their SMS service. This is not surprising given how lucrative the SMS market currently is [17].

<sup>1</sup>Unlimited SMS packages currently do not exist in Canada.

- **Cross traffic:** Similarly, cross traffic within the cellular network may cause excessive loads at the service center. In this study, we are forced to assume that the service provider is sufficiently provisioned to mitigate any effect due to cross traffic.

Unfortunately, the ability to programmatically sample the signal strength and current base station are not available on all mobile devices, nor are they part of the Java CLDC standard [1]. We measure these properties to gain insight; however, incorporating them into our final SMS-TP would violate our CLDC compliance constraint and inhibit the portability of our SMS-TP implementation.

We now describe experiments that isolate the effect that these parameters have on each channel characteristic.

### 3.3.3 Experimentation

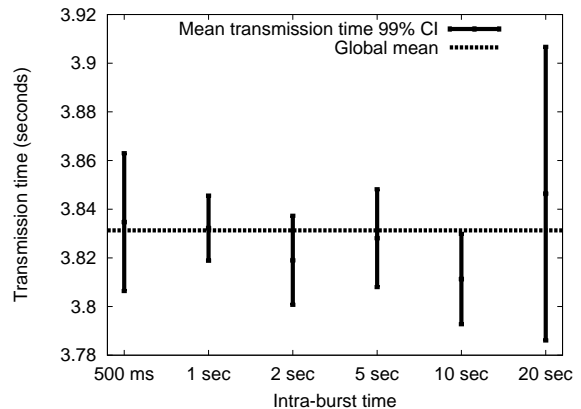
We conduct three experiments that isolate the effect of each parameter on our channel characteristics. In all experiments the sending and receiving devices are located in a low-density urban environment, physically stationary, and stored near an external building window to minimize unnecessary interference. The devices are also separated by approximately five km to guarantee that they are not associated with the same base station and compete for access to the same random access channel. This simple requirement was verified to be satisfied during experimentation.

**Network Variation Experiment:** The first experiment is designed to evaluate the effect of the time-of-day and the network interface used. This experiment consists of four week-long trials. In each trial a different device is used to transmit bursts of 55 messages per hour over the entire week. The value of 55 messages per hour was chosen to create statistically relevant samples while not exceeding the SMS quota on a single SIM card. In this experiment we configure the intra-burst time to be 500 milliseconds. The inter-burst time was configured to be  $3600 - \sum_{i=0}^{55} (t_i + 0.5)$  seconds to ensure messages were transmitted each hour.

**Message Timing Experiment:** Our second experiment examines the effect of intra-burst time by performing several additional week-long trials with varying intra-burst times. In this experiment we evaluate four additional intra-burst times: one, two, five, ten, and twenty seconds. As in the previous experiment, our inter-burst time is an hour minus the time needed to transmit all messages. Each hourly trial in this experiment consists of 55 messages for the same reason. Although in this experiment, we are not explicitly focused on decomposing any effect due to the time-of-day, it is necessary to conduct the trial over the entire week to hold the time-of-day as a fixed variable. Similarly, in this experiment we only use the 8820 devices to remove any effect introduced by the network interface.

**Message Index Experiment:** Our final experiment examines the effect of the transmission index. The previous two experiments were limited to bursts of 55 messages. This experiment consists of four additional trials where 1000 messages are exchanged between a pair of 8820s. This experiment uses the same 500 ms intra-burst time as in the network variation experiment.

Throughout these experiments we assume that the effect caused by time-of-day is independent of the week of the year. While this is an assumption that previous studies have shown to be clearly wrong [11, 22], it is an assumption that we must make to assess the other input variables. To avoid



**Figure 2: Mean transmission time with respect to intra-burst time.**

periods of potential abnormality, we do not perform our experiments over any national holidays or major events.

We also assume that the behaviour of an SMS message is independent of the GSM port used. As of GSM 3.40 [2], SMS messages can be transmitted on one of 65536 ports. Although the Java SMS libraries included with BlackBerry support transmitting SMS messages on different ports, this feature is not common to all mobile platforms. Preliminary experimentation has shown that changing the GSM port has no effect. We therefore exclude the GSM port as an input variable and assume that this result holds for all GSM ports.

## 4. EXPERIMENTAL RESULTS

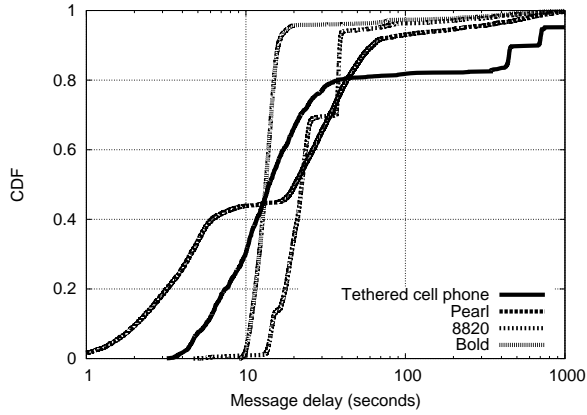
Over the course of our experiments we successfully exchanged 74,990 messages between pairs of cell phones and BlackBerry smartphones. We now present our analysis by considering each of our channel characteristics.

### 4.1 Transmission time

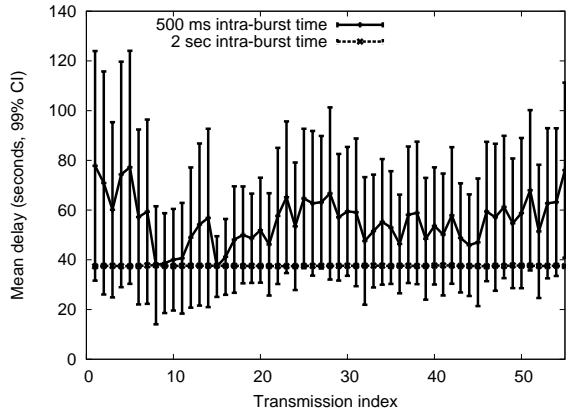
We evaluate this characteristic by considering the effect of each parameter.

**Intra-burst time:** We evaluate the effect of intra-burst time by performing a series of  $t$ -tests that compare each trial's confidence interval (CI) to the sample population mean. The 99% CIs for each intra-burst time are illustrated in Figure 2. Given that each CI contains over 9000 samples, we may ignore the fact that the 10 second test does not contain the population mean and conclude that transmission time is independent of the intra-burst time at a 99% confidence level.

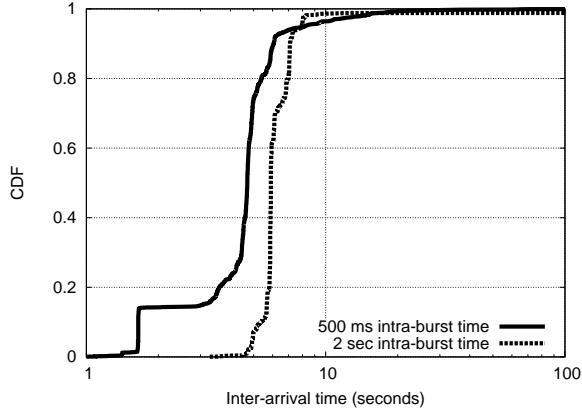
**Transmission index:** We consider the effect of transmission index through a standard one-way analysis of variance that compares the transmission index of a message to the transmission time. In this analysis we assume that transmissions are independent events. The input to this analysis is the week-long 8820 trial from the network variation experiment and the extended 8820 data from the message index experiment. The combination of these two datasets consists of 13,878 SMS messages over approximately 194.45 hours. The independent variable in this analysis is the transmission index, and the dependent variable is the transmission time. For transmission time to be independent of the transmission index, we expect to see more variability within samples



(a) CDF for message delay.



(b) Mean message delay with respect to intra-burst time.



(c) CDF of inter-arrival time (8820 using a 500ms intra-burst time).

**Figure 3: Analysis of delay.**

than between them. Our analysis revealed an F-test statistic of 0.078, which indicates that there is significantly more variability within samples of a particular transmission index than between them and that transmission time is therefore independent of the transmission index. We observed similar results in both the week-long Pearl and Bold datasets.

**Network interface:** The mean transmission time for the Nokia cell phone, Pearl, 8820, and Bold experiments

was 6.02, 4.12, 3.84, and 3.94 seconds respectively. Using a  $t$ -test we found that there was no significant difference between smartphones at a 99% confidence level. The increased time needed to transmit using the cell phone was due to the overhead incurred through communication with the cell phone. Using Nokia’s FBUS protocol, messages are sent asynchronously by first injecting the message into the phone’s outbox. The phone is then polled to verify that the message has been sent. Sending the message took on average 1.5 seconds and a one second polling period added an average delay of 0.5 seconds.

The choice of device did have an effect on transmission failures. Over all smartphone experiments we observed a mean transmission failure rate of 3.07%. The failure rate for the Pearl, 8820, and Bold were 2.93%, 3.10%, and 3.18% respectively; again the rates are statistically equivalent. However, in the cell phone trials we observed a surprisingly low failure rate of 0.7%. The difference stems from the experiment driver’s interface with the cellular hardware. After Gammu injects the message into the phone’s outbox, the phone makes several attempts to establish a dedicated channel and deliver the message. It takes on average 11.14 seconds for the transmission to fail. In contrast, the Java API on the smartphones simply throws an exception to notify the experiment driver that a transmission failed. Detecting a failure on the smartphone takes a mean time of 3.71 seconds.

**Time-of-day:** We observed no significant correlation between the time-of-day and the transmission time. However, we did find that transmission failures are strongly correlated with the time-of-day. Approximately 51% of transmission failures occur during the business hours of 9 am and 6 pm. The remaining failures are uniformly distributed over the rest of the day.

## 4.2 Delay

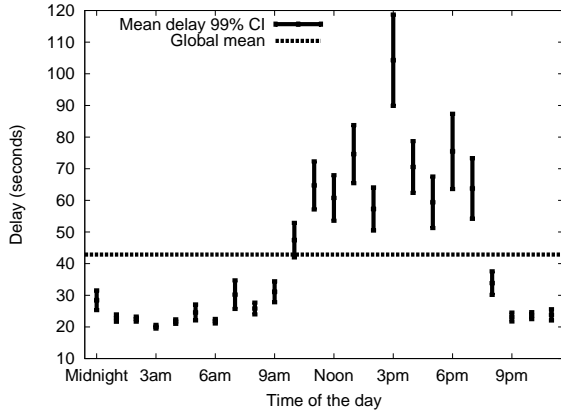
Because SMS messages are delivered sequentially, delay is a characteristic that is tightly coupled with reordering. That is, messages may be delayed purely as a consequence of being reordered and placed in a delivery queue at a later time. Here we examine overall delay. The relationship between delay and reordering is examined in Section 4.4.

Over all the experiments using an intra-burst time of 500 ms, we observed a mean delay of 289.34 seconds in the cell phone trials, and mean delays of 67.23, 39.14, and 25.30 seconds for the Pearl, 8820, and Bold respectively. With over 9,000 samples each, this indicates a significant correlation between delay and the network interface used. These results are illustrated as a delay CDF in Figure 3(a).

We also found that delay was strongly correlated with the intra-burst time and the time-of-day.

**Intra-burst time:** Using the 8820, we found that the delay for messages with a 500 ms and one second intra-burst times is 39.12 seconds. Mean delay falls and stabilizes at 37.57 seconds using an intra-burst time of two seconds. These results are illustrated in Figure 3(b). Although the mean delay of messages with a two second intra-burst time is clearly within the 99% CI of the 500ms case, its low variability indicates a clear difference. Interestingly, we also observe that intra-burst times higher than two seconds offer no improvement in delay.

Given that a service center is capable of delivering multiple messages over one dedicated channel [7], we examine how



**Figure 4: Mean message delay over the day (8820 using a 500ms intra-burst time).**

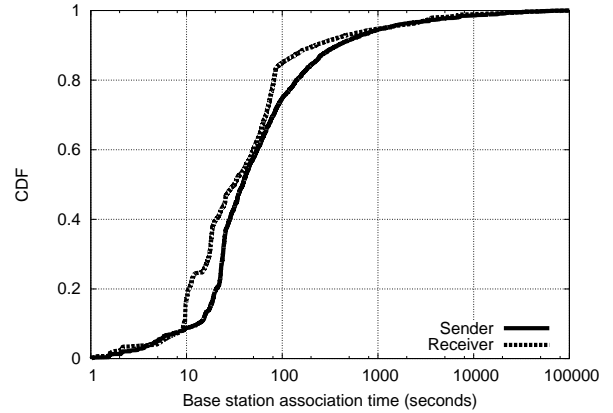
delay impacts the message arrival rate. Surprisingly, despite high delays within the network, messages that are transmitted back-to-back are almost always delivered in batches at a rate of one message every 4 to 6 seconds. We illustrate this finding through the CDF of message arrival time in Figure 3(c). For messages sent with an intra-burst time of 500 ms, the 90 to 98 percentile indicates that the delay between batches of messages is at most approximately three times the mean inter-arrival rate. As we will discuss later in this section, the remaining messages with a high inter-arrival time are due to reordered messages that suffer abnormally high delays. For messages sent with an intra-burst time of 2 seconds, nearly 98% of messages are delivered at a steady rate between one message per 5 to 8 seconds. Although we are unable to determine when a device establishes and releases a dedicated channel, we believe that this is strong evidence that the device takes advantage of the channel to retrieve as many available messages as possible.

**Time-of-day:** Delay varies significantly over the course of a day. Across all trials, we observed that delay is minimized during the hours of midnight to 9 am. On the 8820, mean delay over this period was 42.87 seconds. During the business hours of 9 am to 6 pm mean delay increases to 104.29 seconds and falls to 21.69 seconds during the night. We illustrate the significant variation in mean delay over the day in Figure 4.

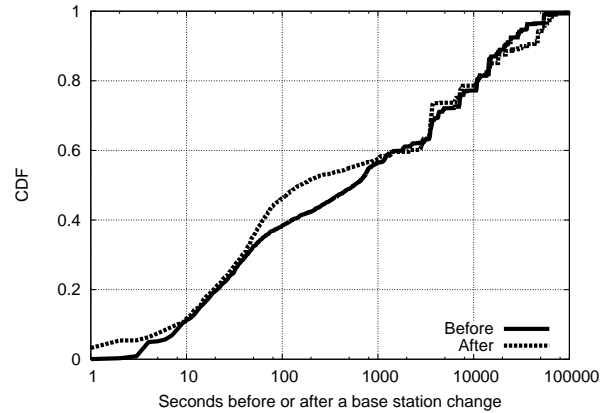
### 4.3 Loss

The loss rate differs significantly between the cell phone and smartphone trials. In the smartphone trials we lost only two messages, which represents a loss rate of approximately 0%. However, the cell phone trials produced a loss rate of 3.89% (approximately 50 messages per day), which is consistent with the results presented in [22]. Moreover, message losses were uniformly distributed over each hour of the day. Although we observed slightly more losses on Friday than on any other day of the week, we found that there is no significant difference between days of the week at a 95% confidence level.

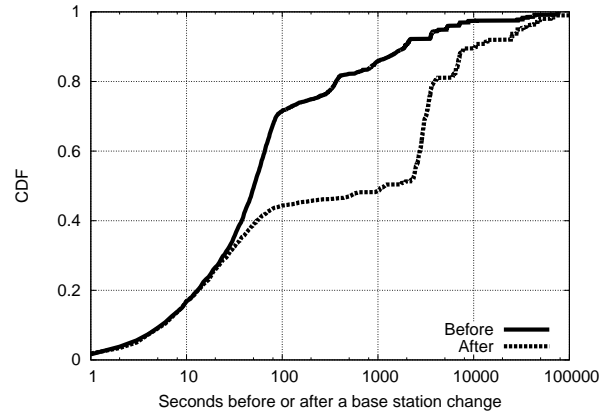
The significant gap between the cell phone and smartphone trials was again due to communication errors with the cell phone. By polling the cell phone for updates, we strongly believe that the device was unable to respond to



(a) Base station association times.



(b) Reordered messages classified relative to base station change. (Sender)



(c) Reordered messages classified relative to base station change. (Receiver)

**Figure 5: Analysis of message reordering.**

paging requests by the cellular network. Then after a series of failed retransmissions, the messages expired within the service center and were deleted. Communication errors with the cell phone also caused 2.91% of messages to be duplicated.

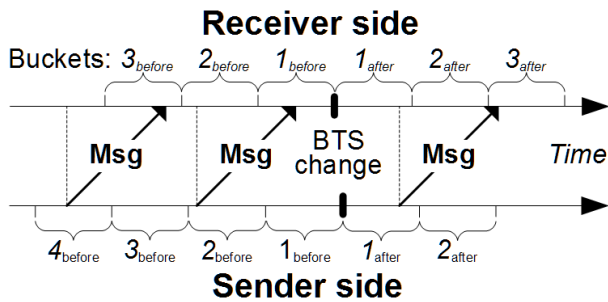


Figure 6: Method for classifying messages with respect to the time of a base station change.

#### 4.4 Reordering

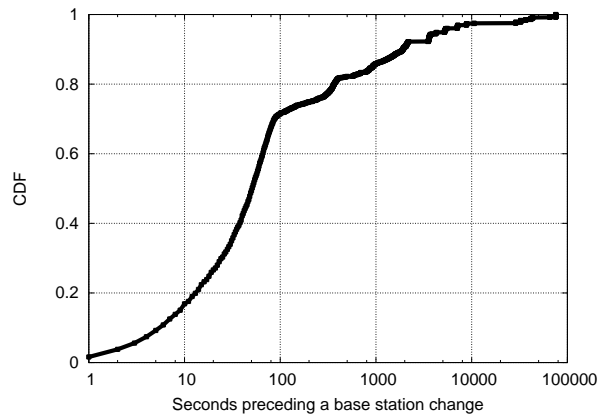
Throughout all trials using an intra-burst time of 500 ms, we observed a global reordering rate of 3.41%. The reordering rate for cell phone, Pearl, 8820, and Bold are 3.01%, 3.73%, 3.07%, and 3.83% respectively. Given so few samples and the low variation between the four devices, we assume that message reordering is independent of the network interface. Our analysis of reordering therefore considers all samples from trials with an intra-burst time of 500 ms.

We found that message reordering is strongly correlated with base station changes and the time-of-day.

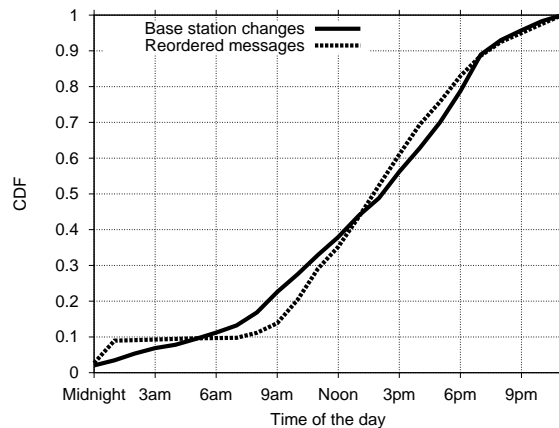
**Base station change:** We classify each reordered message according to the number of seconds before or after a change in base station. For clarity, this classification technique is illustrated in Figure 6. Over the course of our study, we observed 16 unique cell IDs. On average, a device actively sending or receiving messages spends 603.10 seconds associated with a base station before switching to another. The median amount of time that an active device spends on a single cell is 33.71 seconds. We illustrate the CDF for base station association time for both the sender and receiver in Figure 5(a). In this figure we note that the base station association times for each device are nearly identical.

On the sending side of our experiment, we found that there was no correlation between base station change and message reordering. We found that messages were equally likely to be reordered when sent before or after a base station change. This observation is illustrated by nearly identical distributions in Figure 5(b).

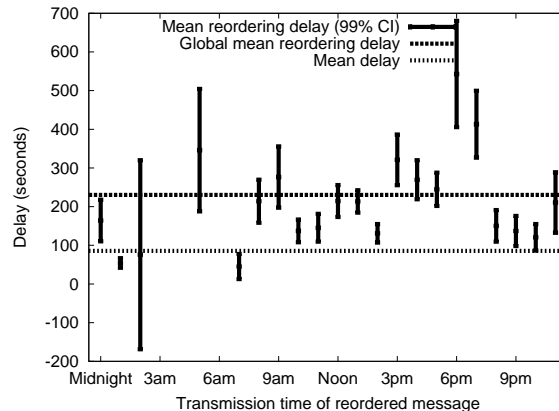
On the receiving side there is a significant correlation between base station change and message reordering. We observe a dramatic difference between the distributions of reorderings. As illustrated in Figure 5(c), nearly 70% of reordered messages are transmitted before the receiver switches to a new base station. Messages that are sent after the receiver makes a base station change are significantly less likely to be reordered. On the receiver we do observe a close relationship between base station association time and the reordering distribution. Figure 5(a) reveals that nearly 80% of base station associations last less than 100 seconds. However, a closer examination of reorderings with respect to when a base station change takes place reveals a non-uniform distribution. Messages that are transmitted closer to the time of a receiver base station change are more likely to be reordered. This distribution is illustrated in Figure 7(a). Interestingly, nearly 50% of the messages that are received out of order were received within the 45 seconds prior to a



(a) Classification of reordered messages with respect to base station change.



(b) Distribution of base station changes.



(c) Mean delay introduced by reordering.

Figure 7: Message reordering and delay.

base station change. Despite a rigorous examination of signal strength and base station change patterns, we have no explanation for this observation; we include it to entice the curiosity of the reader.

**Time-of-day:** Finally, we consider the relationship between the delay introduced by message reordering and time. Over the course of a day, we observe a distinct difference between the frequency of base station changes and the oc-

Key Findings	Design Impact
<i>Transmission time</i>	
The transmission time of SMS messages are independent of the intra-burst time, the transmission index, and the time-of-day.	The protocol does not need to regulate the transmission rate.
<i>Message delay</i>	
Delay is highly correlated with the network interface used and the time-of-day that the message is transmitted.	The protocol must tolerate highly variable delay.
Delay is independent of the transmission index of a message.	The protocol may be agnostic to the quantity of messages transmitted.
Devices utilizes a dedicated channel to retrieve multiple messages from the service center. Despite high delays and reordering, messages are received at a steady rate in batches.	The protocol may assume a relatively constant message arrival rate.
<i>Message loss</i>	
Assuming that both the sending and receiving devices are camped on the cellular network, messages sent from one device to another are lost, in the worst case, 3.89% of the time. Losses that do occur are primarily due to software errors rather than the cellular network.	Message delivery is not guaranteed. The protocol must detect and correct missing fragments of data.
<i>Message reordering</i>	
Messages are reordered at a mean rate of 3.41%. The probability that a message is reordered increases significantly during business hours.	The protocol must be tolerant of a high degree of message reordering.
Message reordering is strongly correlated with base station change.	No impact. Although message reordering could be reduced by actively monitoring signal strength, the ability to programmatically monitor the signal strength of a base station is not available on most mobile platforms [24].

**Table 1: The impact of SMS channel characteristics on design of a transport protocol.**

currence of reordered messages. We illustrate the difference in Figure 7(b). In this figure, we compare the daily CDF of base station hand-overs with the CDF of reordered messages. Clearly, message reordering and its associated delay are not purely dependent on base station changes.

To evaluate *reordering delay* we group reordered messages according to the hour that they were transmitted. Recall that for monotonically increasing message IDs, reordering delay is measured by the difference in time between the arrival of a reordered message  $i$  and the minimum receive time of message  $i + 1, i + 2, \dots$ . Overall we observed a mean reordering delay of 235.98 seconds. This delay is in addition to a mean delay of 99.56 seconds (when the message should have arrived). Therefore a reordering introduces a nearly 240% increase in overall delay. Figure 7(c) illustrates the 99% CIs (two-tailed  $t$  distribution) for each hour of the week. In this figure we immediately see that there is significant difference between the hours of the day. Moreover, by sub-categorizing reordering delay into the three periods of the day, the intra-day variation is even more obvious. We found that 13.3% of reordered messages are transmitted between midnight and 9 am, 57% are transmitted during the hours of 9 am to 6 pm, and the remaining 29.7% occur from 6 pm to midnight. The 99% CIs for the three groups are: (90.27,136.78), (192.49,223.64), and (290.35,398.24) seconds. With relatively large distances between the CIs we can conclude that there is a significant difference in reordering delay between the different periods of the day at a 99% confidence level.

Device	Tx time	Delay	Loss	Reordering
Nokia	6.02 sec	289.34 sec	3.89%	3.01%
Pearl	4.12 sec	67.23 sec	0%	3.73%
8820	3.84 sec	39.14 sec	0%	3.07%
Bold	3.94 sec	25.30 sec	0%	3.83%

**Table 2: Summary of device measurements.**

## 4.5 Summary of results

We briefly summarize the key results of our characterization. Table 2 provides the mean values for each channel characteristic for each test device. Table 1 highlights the key findings of our characterization and how each results impacts the design of the transport protocol.

## 5. TRANSPORT PROTOCOL DESIGN

Although we predict that SMS messages will continue to decrease in monetary cost, the service is not yet free. In addition to the design criteria outlined in Table 1, our design must minimize the *message overhead* needed to exchange data. Minimizing message transmissions also reduces the amount of energy consumed on both the sender and receiver. Finally, our design is further motivated by the concurrent need to maximize data *throughput* over the SMS channel.

In preliminary experimentation we found that bidirectional SMS communication, where devices are simultaneously exchange SMS messages with each other, degrades performance significantly: transmission times and delays are greatly increased, nearly 45% of messages are reordered, and losses are frequent. While this result is a side effect of GSM

channel contention and not a fundamental property of SMS, (and thus not considered in our study), we must minimize bidirectional communication in our protocol design.

## 5.1 Protocol

The communication protocol between two SMS-TP clients is designed to support a wide variety of applications and user defined settings while maximizing the payload of a single message.

### 5.1.1 Flow control and error control

SMS-TP provides flow control and error control through a simplified version of the NETBLT protocol [4]. In NETBLT, the sender communicates with the receiver by exchanging a series of large data aggregates called buffers. When transferring a buffer, the sender fragments it into a set of packets. The packets are then sent across the network to the receiver where they are reassembled into the original buffer. When the last packet in the buffer arrives, the receiver then checks to see if all the packets in the buffer have been correctly received. The receiver then sends an ack to the sender containing a bitmap that indicates the packets that have been received or lost.

NETBLT has several distinct advantages that make it suitable for use in an SMS-based data transport protocol. Bidirectional transmission of SMS messages between devices is minimized through the use of a selective ack that is sent when all packets have been received or a timeout occurs. The selective ack also tolerates message reordering, random losses, and variable inter-arrival times. The low loss rate of SMS ensures that the few acks sent by a NETBLT receiver will almost always be delivered. Finally, NETBLT allows messages to be transmitted in a continuous burst. Given that our findings indicate that messages can be transmitted as rapidly as possible, this attribute of NETBLT significantly simplifies the complexity of our protocol implementation.

Our approach differs from NETBLT by using only one buffer that is bound to 32 KB in size; thus removing the need to coordinate block transfer and to reassemble blocks of data at the receiver. Like NETBLT, buffers are fragmented into SMS message sized *chunks* for transmission to the receiver. The sender initiates communication by sending a single message to the receiver that contains the size of the overall data, an integer representing the current transaction, and the first chunk of data to exchange. Upon receiving the initial message, the receiver may accept or reject the session by sending an ack back to the sender. If either message is lost, the sender will retransmit the original message several times before failing. We refer to the period of time needed for the sender to initiate communication and receive the initial ack as the *setup delay*.

After receiving the initial ack accepting the transaction, the sender begins transferring each remaining chunk of data within the body of an SMS message. When sending each message, the sender adapts to transmission failures by re-sending messages; however, the sender is not aware of losses or reordered messages in the network. The sender continues to transmit chunks of data until none remain.

### 5.1.2 Acknowledgement timers

Each message sent by the sender causes an ack timer to be reset. Because delay is independent of the transmission

---

#### Algorithm 1 SMS-TP algorithm

---

```

intra = START_VALUE
time_recv = time_now
while true do
    event ← waitforevent
    if event = message received then
        add message to receive buffer
        if data fully received then
            send selective ack
            exit {Success}
        else
            if duplicate message then
                send selective ack
            end if
            intra =  $\alpha(intra) + (1 - \alpha)(time_{now} - time_{recv})$ 
            reset_timer(time_now + intra *  $\beta$ )
        end if
        time_recv = time_now
    else if event = timer expired then
        if retry_count > MAX_RETRIES then
            drop partial data
            exit {Failure}
        else
            send selective ack
            retry_count = retry_count + 1
            reset_timer(time_now + last timer value *  $\gamma$ )
        end if
    end if
end while

```

---

index, we know that the first message follows the same delay distribution as subsequent messages. We therefore set the timer duration to be twice the setup delay. Although this value may not be optimal, we found that in practice, a relaxed timer at the sender prevented many redundant message retransmissions. We have also observed that redundant retransmissions arrive after the original message approximately 96% of the time. When the timer expires, the sender begins retransmitting all non-acked chunks.

On the receiving side, the data chunk contained within each received message is placed into a pre-allocated buffer. The transport protocol must then decide whether or not a selective ack should be sent. We propose the following algorithm that exploits our finding that messages arrive at a steady inter-arrival rate in batches. Upon delivery of each message, the receiver computes the exponential moving average of message inter-arrival time. After receiving a message, the receiver sets a timer to  $\beta$  times the inter-arrival time. Given that 98% of messages transmitted back-to-back arrive within three times the mean inter-arrival time, we consider  $\beta = 3$  to be a reasonable selection. Messages that are delayed beyond this threshold are probably lost or subject to abnormally high delays due to reordering.

Upon expiration of the ack timer, the receiver immediately transmits an ack that indicates the chunks that have been received. The receiver then enters an exponential back-off by resetting the timer to  $\gamma$  times its previous value. The choice of  $\gamma$  is less critical than the choice of  $\beta$ . We found  $\gamma = 2$  to be a reasonable selection; however, pathological networks, such as networks that are severely under provisioned, may require a more aggressive back-off rate.

If no messages are received after three ack retransmissions, then the sender is assumed to be gone and the receiver discards the current buffer. This technique allows the receiver to adapt to fluctuations in delay that cause message inter-arrival times to vary. The pseudocode for this algorithm is given in Algorithm 1. As in NETBLT, when the last chunk has been placed in the buffer, the receiver sends a complete ack to the sender. Because this final ack may be lost, the receiver maintains record of the completed data and retransmits completed acks as needed.

## 5.2 Architecture

The software architecture of SMS-TP was primarily influenced by the need for platform portability and code modularity. To run on a variety of resource constrained devices, the architecture minimizes memory allocation and copying by reusing objects, timers, and threads as often as possible. To provide clean integration with existing mobile systems and applications, SMS-TP also provides a series of APIs that abstract platform heterogeneity from the internal operation of the transport protocol.

We encourage the interested reader to see Section 3 in [26] for technical details on the software architecture, implementation, and APIs for third party applications.

## 5.3 Implementation

We have implemented the transport protocol as a library written in Java Micro Edition. Although Java is a portable language, most platforms provide auxiliary libraries to access device specific functionality or provide more memory-efficient data structures [24]. Our implementation is compliant with the Java CLDC and can therefore be embedded into existing applications running on Java enabled cell phones, smartphones, and PC environments. Our implementation can be downloaded for free [8].

## 6. EVALUATION

Our evaluation of SMS-TP consists of two stages. First, we evaluate the performance of the transport protocol using two sample Java applications that exchange data between a pair of SMS-TP enabled devices. This evaluation mimics the behaviour of a real deployment. The second stage of our evaluation simulates SMS service using a range of delay and loss rates.

### 6.1 Sample applications

Our first application is written in standard Java for Linux and the second is written in for BlackBerry. We use both the cell phone and BlackBerry 8820 to conduct our evaluation. Both devices were configured the same as in our previous experiments. For measurement purposes, all devices' clocks are synchronized to either an NTP server or the cellular network. Each application operates as either sending or receiving mode. The sending application generates random data and sends the data to the receiver using the SMS-TP library; giving the receiving device's phone number as the destination address. The receiving application blocks waiting for data from SMS-TP. Upon receiving data, the data is written to a file.

We evaluate SMS-TP by exchanging data in repeated intervals of 512 bytes, 1 KB, 2 KB, 4 KB, 8 KB, 16 KB, and 32 KB. The results are compared to a calculated optimal case. In the optimal case, we simulate the transmission of each

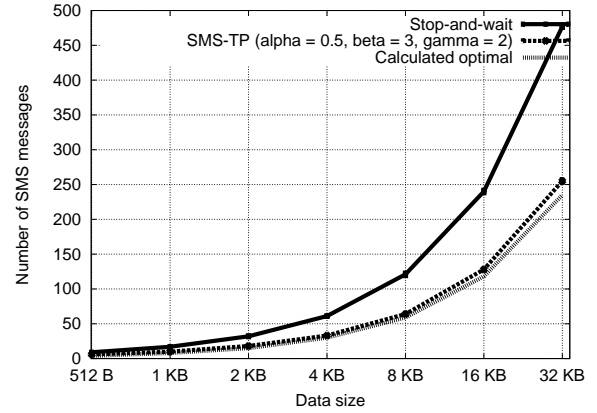


Figure 8: Message overhead.

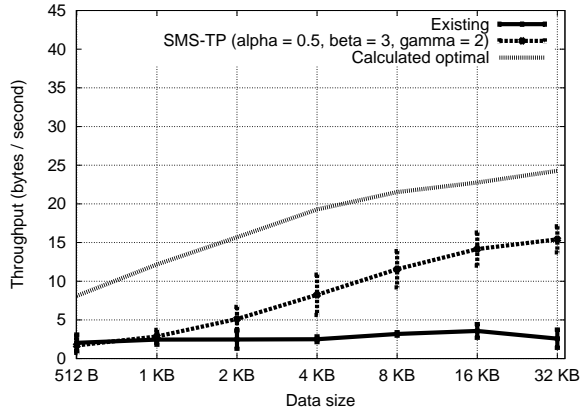
data bundle under perfect conditions. We use the constant mean transmission time and delay, with no losses, no re-ordered messages, and no transmission failures. Given these generous assumptions, it is not necessary to use SMS-TP: the optimal case does not require a header on each message and it can fragment data into full 140 byte SMS messages. The optimal case maximizes its use of pipelining to transfer messages as fast as possible. Our evaluation also includes an implementation of the approach used by existing applications, which we will describe next.

We conduct our evaluation during the hours of midnight to 6 am for consistency in our measurements. Our evaluation is focused on two key variables: message overhead and data throughput. When measuring throughput, we assume that the transaction is over when the receiver finishes writing the data to file, and not when the receiver receives the final ack.

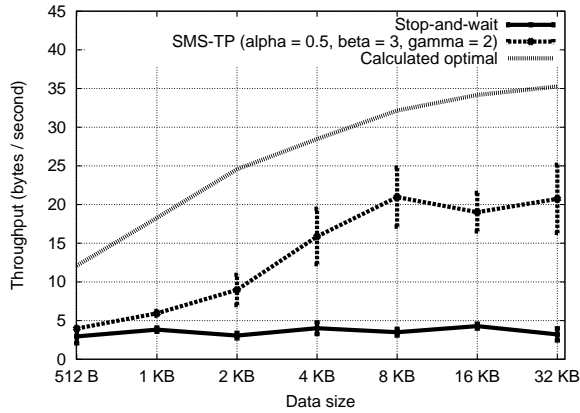
#### 6.1.1 Existing method

The method used by existing mobile applications to exchange data over SMS is both simple and reliable. Applications that must exchange more than 140 bytes of data do so by fragmenting their data and sending it using a series of messages [35, 33]. Because losses are known to occur in the cellular network [39], applications achieve reliable data transport by utilizing a per-message ack. After sending each message, the sender initiates a ack timer and blocks for a delivery receipt (ack) before sending the next message. If the timer expires before the ack is received, then the message is retransmitted. In our evaluation, we set the ack timeout to 160 seconds (approximately twice the measured RTT). Alternatively, the sender may utilize the delivery receipt mechanism in SMS to receive confirmation of delivery from the service center instead of the receiving device. The sender continues to send messages in series until the data is fully delivered. The one-to-one relationship between messages and acks negates the use of timers at the receiver. This protocol is therefore very simple to implement.

In this protocol, each message includes a small, fixed sized header that allows the fragments to be reassembled at the receiver. This header includes both the total size of the data and the sequence number of the current fragment. In our implementation, we represent the size and sequence number as two bytes each. The remaining 136 bytes of the message



(a) Throughput using a tethered cell phone.



(b) Throughput using a smartphone.

**Figure 9: Message overhead and throughput analysis of SMS-TP.**

are always used, with the exception of the last message, which is not required to use the entire 136 byte payload.

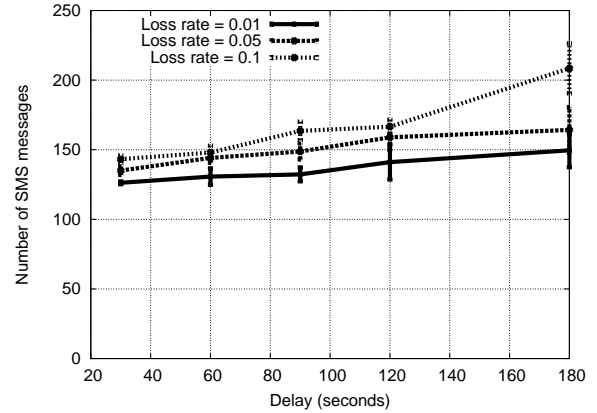
Although this method is both easy to implement and reliable, it is inefficient. This method at least doubles the number of messages required to transfer the data and significantly increases the time needed to transfer the underlying data.

### 6.1.2 Message overhead

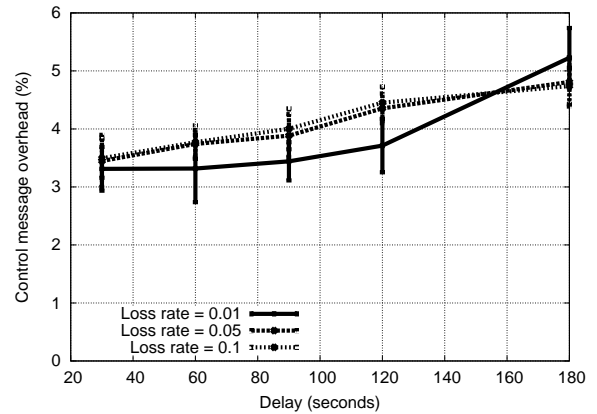
The message overhead (99% CI) for transferring each data size for each of the three cases using the 8820 is illustrated in Figure 8. We found that the overhead using the cell phone was statistically equivalent. As expected, we found that SMS-TP offered a nearly 50% gain over the existing method. This gain is due to our consolidation of acknowledgement into a single selective ack sent when the receiver has received all of the messages, or a timeout occurs. There were no retransmissions as a result of ack timeouts in our evaluation of the existing method. In the worse case, the message overhead of SMS-TP was only 8.5% below the optimal case.

### 6.1.3 Throughput

We found that SMS-TP yields a significant improvement in data throughput over the existing method. We illus-



(a) Message overhead for SMS-TP for variable loss rate and delay.

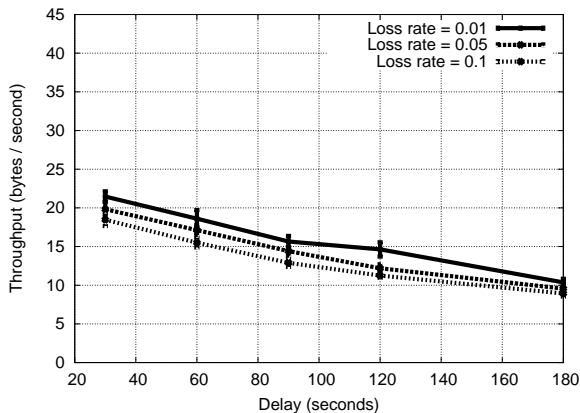


(b) Control message overhead.

**Figure 10: Message overhead.**

trate the results of our evaluation for both the cell phone and smartphone in Figure 9(a) and Figure 9(b) respectively. Using the cell phones, we found that SMS-TP improves throughput by as much as 499% when transferring 32 KB of data. Using the smartphone, we found that throughput leveled off when transferring 8 or more KB of data. Using the smartphones, our protocol improves throughput over SMS by as much as 545%! The significant gap in throughput between the cell phones and smartphones is primarily due to frequent communication failures with the cell phone that inhibit the phone's ability to send and receive messages and deliver messages to the PC over the serial connection. It is a testament to SMS-TP's design that such throughput gains can be made in the presence of frequent failures; we expect that in the absence of these communication failures, data throughput using a tethered cell phone would improve dramatically.

We also found that in the worse case SMS-TP lags behind the optimal case by approximately 40%. Given the generosity of the assumptions made in the optimal case, we believe that this is a good first step. Narrowing this gap (if possible) will be an area of future study.



**Figure 11: Throughput for SMS-TP for variable loss rate and delay.**

## 6.2 Simulation

In the second phase of our study we examine the effect of alternate loss rates and delay on the message overhead and the throughput of SMS-TP through simulation. We select alternate characteristics from within the 90 percentile of values observed by Zerfos in [39]. Our simulation therefore examines the performance of SMS-TP when deployed in a developing region.

Our simulation consists of two unmodified instances of SMS-TP, each initialized with an SMS Handler that simulates both the device and cellular network. Each message enqueued for transmission by an SMS-TP instance is subject to transmission and propagation delays using specified mean and standard deviation values. Messages are also discarded by the simulator as a configurable loss rate. Messages that are not discarded are delivered to the destination SMS-TP instance, which can respond using the same simulated channel. The simulation library is included as part of SMS-TP distribution [8].

Our evaluation consists of three loss rates: 1%, 5%, and 10%. Delay also varies between 30 seconds to 180 seconds. In each trial we use a mean transmission time of 3.8 seconds and exchange 16 KB of random data.

### 6.2.1 Message overhead

The mean total message overhead is illustrated in Figure 10(a) (95% CI). Under a 1% loss rate and low delay, the simulation is statistically equivalent to our previous results. Under low-loss and high-delay conditions, message overhead increases by less than 10%. Under high-loss and high-delay conditions, the receiving instance of SMS-TP will occasionally send a premature ack, which causes the sender to retransmit copies of messages that may still be in transit. The increase in acks (control messages) as a percentage of total message overhead is illustrated in Figure 10(b).

### 6.2.2 Throughput

Similarly, under low-loss, low-delay conditions, the simulation yields similar throughput: approximately 20 bytes / second for our 16 KB trial. These results are illustrated in Figure 11. The pipelined approach to message transmission mitigates the effect of message losses on throughput. Even under high-loss conditions, throughput remains within 20%

of the low-loss throughput boundary. When subject to high delays throughput falls to approximately 10 bytes / second and message losses have negligible effect on throughput. Interestingly, under high-loss, high-delay conditions, SMS-TP outperforms the existing method, evaluated under low-loss, low-delay conditions, by a factor of two.

## 7. CONCLUSION

We have presented the successful design and implementation of an efficient and reliable SMS-based data transport protocol. We motivated the design of our transport protocol through an empirical study that characterizes the behaviour of the cellular network under bursty workloads. This study complements existing SMS studies by considering factors that affect SMS: intra-burst time, transmission order, the network interface used, time-of-day, signal strength, and base station changes. To our knowledge, it is the first paper that measures SMS from the perspective of mobile applications using the service. Among many interesting results, we were surprised to find that the cellular network does not inhibit rapid transmission of SMS messages, message delay varies significantly over the day, and that messages transmitted back-to-back are almost always delivered in batches at a steady rate. Although our findings are based on one network and a limited number of devices, preliminary experimentation on GSM networks in both the United States and Europe suggest that other networks and devices will exhibit similar properties. As the first study to examine SMS from the applications' perspective, rather than from within a service provider's network, we believe that our work will serve as a basis for future study. Using our publicly available measurement tools and scripts, we plan to expand our dataset to include additional cellular networks in other countries.

The transport protocol has been implemented as a stand-alone Java ME library that is CLDC compliant and runs on a wide range of cell phones, smartphones, and PC environments. Our implementation has a memory footprint of only 33 KB and can be easily integrated with existing mobile system. Using this implementation, we show that the transport protocol outperforms existing techniques by reducing message overhead by as much as 50% and by increasing data throughput by as much as 545% over the approach used by existing mobile applications.

We believe that this work has the greatest potential impact in developing regions where alternatives to SMS are either expensive, poorly deployed, or do not exist. Our SMS-TP implementation is currently used to provide high-priority, moderate-latency communication to rural kiosks in the VLink delay-tolerant networking system [38], as a control channel in the MobiTether middleware [25], and to exchange cryptographic information between mobile devices in NearbyFriend [40]. We believe that many other mobile systems, both in the developed and developing world, could benefit from this work.

## 8. ACKNOWLEDGEMENTS

My thanks to Hossein Falaki, Dr. Pan Hui, Prof. Aaditheshwar Seth, Prof. Urs Hengartner, Prof. Philip Levis, and Prof. S. Keshav for their feedback on this paper.

This research was supported by grants from the National Science and Engineering Council of Canada, the Canada Research Chair Program, Cisco Research, and Nokia Research.

## 9. REFERENCES

- [1] JSR 120. Last visited 04/04/2010. <http://jcp.org/en/jsr/detail?id=120>.
- [2] 3GPP Specification. Last visited 12/10/2008. <http://www.3gpp.org/ftp/Specs/html-info/23040.htm>.
- [3] Jay Chen, Brendan Linn, and Lakshminarayanan Subramanian. Sms-based contextual web search. In *MobiHeld '09: Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*, pages 19–24, New York, NY, USA, 2009. ACM.
- [4] D. D. Clark, M. L. Lambert, and L. Zhang. Netblt: a high throughput transport protocol. *SIGCOMM CCR*, 17(5):353–359, 1987.
- [5] CTIA. The Wireless Association Announces Semi-Annual Wireless Industry Survey Results. Last visited 19/06/2009. <http://www.ctia.org/media/press/body.cfm/prid/1811>.
- [6] William Enck, Patrick Traynor, Patrick McDaniel, and Thomas La Porta. Exploiting open functionality in sms-capable cellular networks. In *Proceedings of CCS 2005*, pages 393–404, New York, NY, USA, 2005. ACM.
- [7] ETSI. Technical realization of the short message service. TS 100 901, GSM 03.40 version 6.1.0, July 1998.
- [8] SMS-TP Files. Last visited 23/03/2010. <http://blizzard.cs.uwaterloo.ca/eaoliver/sms.html>.
- [9] Gammu. Last visited 16/06/2009. <http://gammu.org>.
- [10] Majid Ghaderi and Srinivasan Keshav. Multimedia messaging service: System description and performance analysis. In *Proceedings of WICON 2005*, pages 198–205, Washington, DC, USA, 2005. IEEE Computer Society.
- [11] Acision. Huge global growth in SMS continues over the new year period. Last visited 16/06/2009. <http://www.acision.com/>.
- [12] GSM World. GSM Coverage Maps. Last visited 12/10/2008.
- [13] Yieh-Ran Haung and Jan-Ming Ho. Overload control for short message transfer in gprs/umts networks. *Inf. Sci. Inf. Comput. Sci.*, 170(2-4):235–249, 2005.
- [14] Y.R. Haung. Determining the optimal buffer size for short message transfer in a heterogeneous GPRS/UMTS network. *Vehicular Technology, IEEE Transactions on*, 52(1):216–225, 2003.
- [15] Rogers Communications Inc. Last visited 13/12/2009. <http://www.rogers.com>.
- [16] S. Keshav. Why cell phones will dominate the future internet. *SIGCOMM CCR*, 35(2):83–86, 2005.
- [17] S. Keshav. The cost of text messaging. In *Cell Phone Text Messaging Rate Increases and the State of Competition in the Wireless Market*. U.S. Senate Subcommittee on Antitrust, Competition Policy and Consumer Rights, June 2009.
- [18] Mugo Kibati and Donyaprueth Krairit. The wireless local loop in developing regions. *Commun. ACM*, 42(6):60–66, 1999.
- [19] G. Le Bodic. *Mobile messaging technologies and services: SMS, EMS and MMS*. John Wiley & Sons Inc, 2005.
- [20] Mark Wood. SMS bulk messaging, the problem and the solution. Last visited 12/10/2008. [http://www.ceasa-int.org/library/7\\_sms\\_mass\\_messaging\\_problems\\_V1-2.doc](http://www.ceasa-int.org/library/7_sms_mass_messaging_problems_V1-2.doc).
- [21] Shengguang Meng, Wei Chen, Gang Liu, Shitong Wang, and Liu Wenyin. An asset management system based on rfid, webgis and sms. In *ICUIMC '08: Proceedings of the 2nd international conference on Ubiquitous information management and communication*, pages 82–86, New York, NY, USA, 2008. ACM.
- [22] X. Meng, P. Zerfos, V. Samanta, SHY Wong, and S. Lu. Analysis of the Reliability of a Nationwide Short Message Service. *Proceedings of INFOCOM 2007*, pages 1811–1819, 2007.
- [23] Zohar Naor. An efficient short messages transmission in cellular networks. In *Proceedings of INFOCOM 2004*, pages 640–649, 2004.
- [24] Earl Oliver. A survey of platforms for mobile networks research. *SIGMOBILE Mobile Computer Communication Review*, 12(4):56–63, 2008.
- [25] Earl Oliver. Enabling the Next Frontier in Mobile Applications. *ExtremeCom 2009*, August 2009.
- [26] Earl A. Oliver. Exploiting the short message service as a control channel in challenged network environments. In *Proceedings of CHANTS 2008*, pages 57–64, New York, NY, USA, 2008. ACM.
- [27] V. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California at Berkeley, Berkeley, CA, USA, 1998.
- [28] C. Peersman, S. Cvetkovic, P. Griffiths, and H. Spear. The global system for mobile communications short message service. *IEEE Personal Communications*, 7(3):15–23, 2000.
- [29] Poria Research. Mobile Messaging Futures 2009–2013. Last visited 19/06/2009. <http://www.portioresearch.com/MMF09-13.html>.
- [30] RIM. Last visited 16/06/2009. <http://www.rim.com>.
- [31] A.N. Rosenberg and S. Kemp. *CDMA capacity and quality optimization*. McGraw-Hill Professional, 2003.
- [32] Jochen H. Schiller. *Mobile communications*. Addison-Wesley, Reading, MA, USA, second edition, 2003.
- [33] Mohammad Shirali-Shahreza and Sajad Shirali-Shahreza. Sending pictures by sms. In *ICACT'09: Proceedings of the 11th international conference on Advanced Communication Technology*, pages 222–223, Piscataway, NJ, USA, 2009. IEEE Press.
- [34] Mukda Suktarachan, Patthrawan Rattanamanee, and Asanee Kawtrakul. The development of a question-answering services system for the farmer through sms: query analysis. In *KRAQ '09: Proceedings of the 2009 Workshop on Knowledge and Reasoning for Answering Questions*, pages 3–10, Morristown, NJ, USA, 2009. Association for Computational Linguistics.
- [35] Ashraf A. Tahat. Implementation of an sms-based telemedicine system for patient electrocardiogram monitoring. In *Telehealth/AT '08: Proceedings of the IASTED International Conference on*

- Telehealth/Assistive Technologies*, pages 223–228, Anaheim, CA, USA, 2008. ACTA Press.
- [36] Patrick Traynor, William Enck, Patrick McDaniel, and Thomas La Porta. Mitigating attacks on open functionality in sms-capable cellular networks. In *Proceedings of MobiCom 2006*, pages 182–193, New York, NY, USA, 2006. ACM.
- [37] Pauliina Tuomi. Sms-based human-hosted interactive tv in finland. In *UXTV '08: Proceeding of the 1st international conference on Designing interactive user experiences for TV and video*, pages 67–70, New York, NY, USA, 2008. ACM.
- [38] VLink. Last visited 12/06/2009.  
<http://blizzard.cs.uwaterloo.ca/vlink/>.
- [39] Petros Zerfos, Xiaoqiao Meng, Starsky H.Y Wong, Vidyut Samanta, and Songwu Lu. A study of the short message service of a nationwide cellular network. In *Proceedings of IMC 2006*, pages 263–268, New York, NY, USA, 2006. ACM.
- [40] G. Zhong, I. Goldberg, and U. Hengartner. Louis, Lester and Pierre: Three Protocols for Location Privacy. *LNCS*, 4776:62, 2007.