Enabling Censorship Tolerant Networking

by

Earl Oliver

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2012

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Billions of people in the world live under heavy information censorship. We propose a new class of delay tolerant network (DTN), known as a censorship tolerant network (CTN), to counter the growing practice of Internet-based censorship. CTNs should provide strict guarantees on the privacy of both information shared within the network and the identities of network participants. CTN software needs to be publicly available as open source software and run on personal mobile devices with real-world computational, storage, and energy constraints. We show that these simple assumptions and system constraints have a non-obvious impact on the design and implementation of CTNs, and serve to differentiate our system design from previous work.

We design data routing within a CTN using a new paradigm: one where nodes operate selfishly to maximize their own utility, make decisions based only on their own observations, and only communicate with nodes they trust. We introduce the Laissez-faire framework, an incentivized approach to CTN routing. Laissez-faire does not mandate any specific routing protocol, but requires that each node implement tit-for-tat by keeping track of the data exchanged with other trusted nodes. We propose several strategies for valuing and retrieving content within a CTN. We build a prototype BlackBerry implementation and conduct both controlled lab and field trials, and show how each strategy adapts to different network conditions. We further demonstrate that, unlike existing approaches to routing, Laissez-faire prevents free-riding.

We build an efficient and reliable data transport protocol on top of the Short Message Service (SMS) to serve a control channel for the CTN. We conduct a series of experiments to characterise 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.

A CTN's dependency on opportunistic communication imposes a significant burden on smartphone energy resources. We conduct a large-scale user study to measure the energy consumption characteristics of 20,100 smartphone users. Our dataset is two orders of magnitude larger than any previous work. We use this dataset to build the *Energy Emulation*

Toolkit (EET) that allows developers to evaluate the energy consumption requirements of their applications against real users' energy traces. The EET computes the successful execution rate of energy-intensive applications across all users, specific devices, and specific smartphone user-types. We also consider active adaptation to energy constraints. By classifying smartphone users based on their charging characteristics we demonstrate that energy level can be predicted within 72% accuracy a full day in advance, and through an *Energy Management Oracle* energy intensive applications, such as CTNs, can adapt their execution to maintain the operation of the host device.

Acknowledgements

One of the dominant themes influencing the design of this work is *selfishness*, coupled with the expectation that resource-constrained entities will expend resources in a manner that maximizes their own utility. However, to paint a picture of almost palpable irony, this work would not have come to fruition without a significant degree of *altruism* by my advisor S. Keshav. I am, by nurture, a difficult person to advise on anything. However, I credit Keshav for successfully passing on the scientific tradition that made this work possible. I am also grateful to Keshav for tolerating my *many* intellectual curiosities and entrepreneurial 'distractions' that were necessary to achieve my goals. Without them, the opportunity cost of this endeavour would have been too great.

Thank-you to the members of the Tetherless Computing Lab, past and present, for being great lab mates and friends. In particular, in alphabetical order, Aaditeshwar, Andy, Gail, Hossein, Omid, Rayman, Ryan, and Tommy helped me out over the years. The five months that I spent at the Technicolor (formerly Thomson) Paris Research Lab at the beginning of my Ph.D. was instrumental in shaping my research interests. Thank-you to Christophe Diot and Jon Crowcroft for many spirited and colourful discussions, and thanks to my colleague and friend Anna-Kaisa, who ended up being my only collaborator. A Ph.D. is a long, hard, mental and emotional haul; my extra-lab friends, Adriana, Ardeshir, Bryanne, Carol, Cristina, Kristen, Nika, Nilam, Yvonne, and Zach were also instrumental to maintaining a steady, stable, mental state. Between lab mates and friends, it was rare not to have someone to share a coffee with. I also thank my family for all the good advice, opportunities, encouragement, and unconditional support that they have given me all these years. I would not be here without them.

Thank-you to Research in Motion, Ltd. for their in-kind donation of BlackBerry devices. Although the devices were donated for teaching purposes, they became invaluable to developing, testing, and evaluating my work. I would also like to thank the students, faculty, and staff that participated in the field trial of my Censorship Tolerant Networking system. Their dedication and active support for my work was much appreciated. The many anonymous participants of my energy consumption study were also essential to my work.

I would like to recognize and thank my fellow Canadian taxpayers for the financial support that I received over the course of my Ph.D. As a recipient of the people's money,

I believe that I have a duty to provide commensurate value to society. I sincerely hope that those Canadians reading this feel that their $0.003 contribution to my work, though unbeknownst to them, went to a worthy cause.

I would be remiss if I did not, perversely, thank the useful-idiot, Internet censors that have already blocked my research website within their country, and are about to block this thesis. Without their hard work and dedication, my thesis topic would not have been so relevant to the world.

Dedication

To my best friend.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Freedom is the right of all sentient beings.

— Optimus Prime

1.1 Introduction

Billions of people in the world live under heavy information censorship [40, 150, 194]. Countries such as China, Saudi Arabia, and Iran practice strict filtering of Internet content [22]. These countries are also quick to tighten such filters during periods of social unrest. China in particular continues to reinforce its "Great Firewall of China" and is working to remove all anonymity from Internet and cell phone usage [8, 88]. In Iran, blogging against the government can come with a penalty of over 19 years [189]. Uzbekistan, Syria, Morocco, Vietnam, among others, have enhanced their censorship to stifle the echoes of the revolutions agitating the Arab world [17, 170, 178, 198].

Currently, one out of every three Internet users is unable to access the free Internet. Net censorship is evolving from a practice of repressive regimes to the norm. Approximately 60 countries are implementing some form of Internet censorship, which entails either content filtering or harassment of citizens based on their Internet usage behaviour [89]. Figure 1.1 illustrates the world map of censorship.

Other countries are expected to begin the practice in the months and years to come. For the first time, Bangladesh has blocked access to certain sites because of videos deemed offensive to their religion [152]. Cambodia is censoring news sites that are critical of the Cambodian ruling party [79]. Even India is moving beyond the censorship of 'obscenity' and beginning to target social networks to remove content deemed 'offensive' [165].

Several software systems have emerged to resist the growth of censorship [38, 44, 55, 190, 191]. These communication systems are often highly distributed to provide both *robustness* to attack and *scale* to large numbers of users, and utilize some form of public key infrastructure (PKI) to provide communication *privacy* [155].

Unfortunately, even these systems can be easily blocked by governments due to their use Internet-based infrastructure [27]. Directed by a repressive government, Internet service providers can easily block access to the IP addresses of anti-censorship services. It is also conceivable that in the future repressive regimes may regard using censorship-evasive

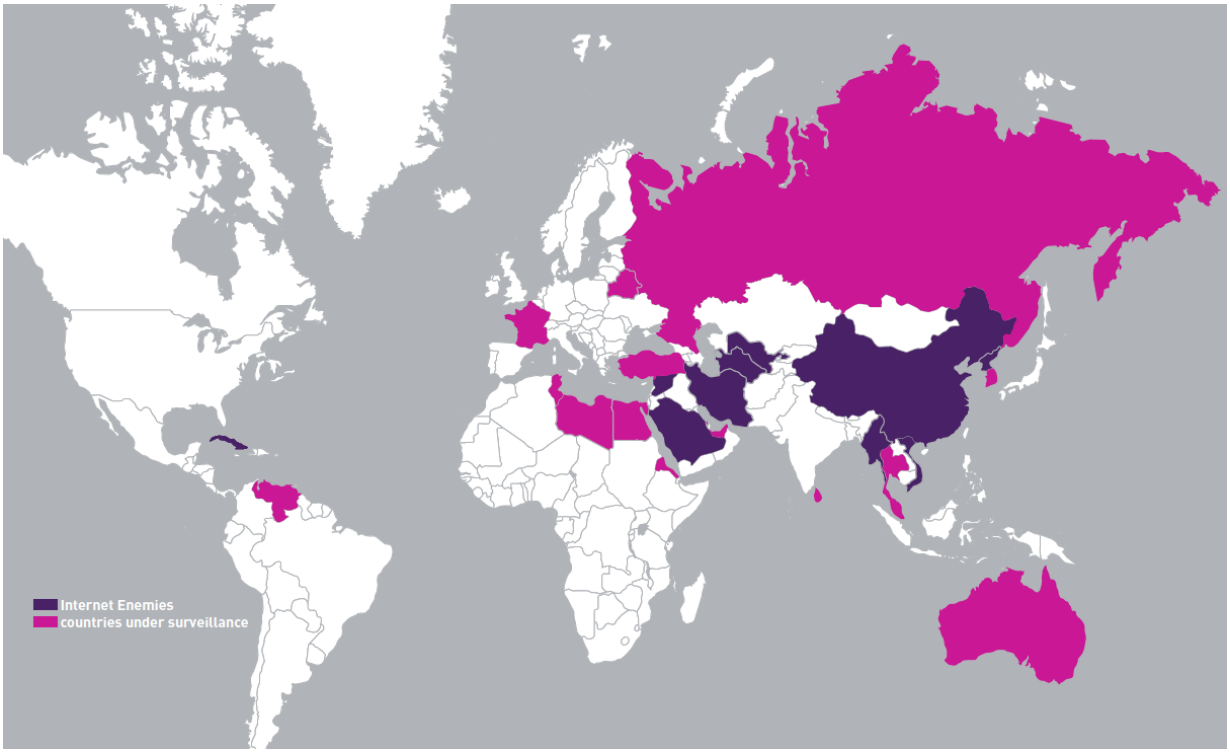


Figure 1.1: World map of censorship. Image courtesy of Reporters Without Borders [150].

systems as an offence in itself. We therefore introduce, in addition to robustness, scalability, and privacy, a fourth tenet of censorship resistance: *unlinkability*. Unlinkability requires that no central authority is able to determine that two entities are communicating with each other [155]. This definition of unlinkability is stronger than the definition of unlinkability commonly used in security literature [140]. The weaker definition requires that no central authority is able to determine that two end-points are communicating with each other. The Tor system, for example, which routes messages through internationally dispersed onion routers, satisfies this weaker definition [44].

Not surprisingly, strong unlinkability is impossible to achieve in Internet-based systems due to their inherent use of infrastructure, which can easily be monitored by the government or a sufficiently compliant Internet service provider [154]. However, mobile ad hoc networks trivially achieve unlinkability due to the fact that any two nodes can create a network anywhere and at any time. Robustness and privacy are also trivially achieved. However,

mobile ad hoc networks trade-off scalability for unlinkability, and are only practical for small groups of users.

We claim that the only way to achieve complete censorship resistance is through a delay tolerant network (DTN), where geographically dispersed personal mobile devices are used to disseminate censored content to interested parties. However, to the best of our knowledge, censorship resistance has never been considered in DTNs.

This thesis proposes a new class of DTN that we call a *censorship tolerant network* (CTN) that strictly adheres to the four tenets of censorship resistance. While DTNs have conventionally been designed to provide connectivity to the geographical fringes of the world [45, 66, 85, 137], CTNs are targeted for average smartphone users with real-world resource constraints.

1.2 Censorship Tolerant Networking

DTNs have conventionally been designed to provide network connectivity to developing regions [25]. These networks are typically owned and operated by a single entity, such as a non-government organization (NGO) or a research group, which has the authority to regulate the operational behaviour of the network's constituent nodes. In these networks, nodes operate according to prescribed protocols to maximize the aggregate utility of the network. Attempts have been made to create DTNs for average smartphone users [144, 176]; however, these networks continue to force nodes to selflessly, and therefore, in our opinion, irrationally, expend limited resource for the sole benefit of others.

Given the resource constraints of personal mobile devices [129], (energy being the most significant constraint [52, 130]), resources must be rationed. We believe that rational users are unlikely to expend limited resource to carry messages on behalf of other users without a future expected benefit, even if the resources are used to disseminate censored, valuable content. As designers of an autonomous mobile system, the assumption that users are selfish and do not want their devices' resources depleted unfairly has a significant influence on our system design. However, as we will discuss later in this chapter, users that are less sensitive to resource expenditures may simply ignore mechanisms to ensure fairness and expend resources as they see wish.

A CTN must satisfy both the strict criteria of censorship resistance: robustness, scalability, privacy, and unlinkability [155], and respect the needs of rational users. This introduces the following design constraints:

- **Identity privacy**: The network must ensure that no non-trusted third party is able to determine the contents of a message exchanged between two mutually trusted nodes. Moreover, a node cannot learn the identity of another node unless they mutually trust each other.
- **Infrastructure-independence**: The network must operate independent of communication infrastructure and servers on the Internet. However, communication infrastructure may be used for direct node-to-node communication to reduce delay in retrieving content and improve network QoS. This requirement implicitly provides robustness and unlinkability.
- **Openness**: The effectiveness of a CTN is proportional to node density. To encourage deployment, the operation of the network must be transparent to all users, including free-riders. We consider the ability to distribute the source code without negative consequences to be a necessary condition of the network's openness. In Chapter 2 we show that prior DTN work is ill-equipped to operate under this design constraint.
- **No specified forwarding protocol**: As a consequence of *openness*, nodes must be free to allocate/expend resources and forward messages in a manner that maximizes their own utility. Nodes within a CTN are only be required to implement a standard pairwise tit-for-tat protocol.
- **Deployability**: The network must be incrementally deployable and not require a central authority to authorize membership or verify identity.
- **Participation fluidity**: The network must allow nodes to enter or exit the CTN at any time. Independent CTNs should be able to merge or separate as participation in the network fluctuates.

1.2.1 Use cases

We envision two main use cases for this work.

The technical savvy activist

Internet censorship[1] is a continual arms race between governments blocking content and those that pursue it. Fortunately, there exist citizens of many countries that are able to circumvent government censorship [8, 150] and download content to their computer and subsequently their personal mobile device. In the pursuit of spreading censored content through a country, we believe that these users will be the proverbial 'patient zero'.

Imagine the following scenario. A technical savvy individual in Iran, which we'll call Casper, connects to VPN server outside of the country on a daily basis to download the daily news from `balatarin.com` (a popular Persian blog that is currently blocked by Iran due to its role in pro-democracy movements [134]). While visiting the site, Casper downloads a compressed archive of the daily news onto their computer and transfers it to his mobile device.

Casper has many friends both at work and in the neighbourhood. On the way to work Casper stops at a coffee shop frequented by friends. These friends are trusted and share the same pro-democracy interests. While sharing a coffee, Casper's device detects the smartphones of his friends, and connects to each of them. His friends' devices vibrate signaling that new content has been discovered. Upon examination of their devices they see that Casper's device contains the latest digest of Balatarin! As avid readers, they all select the item for download. While Casper's device is connected to one of his friends' devices, the content is transferred to them. He then leaves causing a subsequent download to be broken. His friends, who trust each other, continue downloading the content from each other while they continue to be collocated in the coffee shop.

At work, Casper only trusts a select few coworkers, none of which desire political news. Upon arrival these coworker quickly learn of Casper's news, but do not download the content. One of them happens to be well connected and trusts many coworkers that do

[1] Internet censorship may be imposed throughout the Internet, from tier 1 ISPs down to home gateways. However, the term generally refers to state-mandated Internet censorship.

desire political content. Upon discovery of fresh political content, they want it! However, since Casper does not know who they are, and obviously does not trust them, he cannot deliver the desire content to them. Instead, the intermediate mutually trusted coworker retrieves the content from Casper, stores it, and eventually delivers it to the requesting coworker. Through this process, both the trusted intermediate coworker and requester possess a copy of the censored content.

We envision that this daily process could be used to disseminate enormous quantities of censored content throughout a population.

Viral content

Restricting a country's Internet and cellular communication infrastructure during periods of civil unrest is an effective means of inhibiting a population from organizing protests and other forms of social disobedience [154]. Shutting down communication infrastructure further prevents citizens from uploading inflammatory (to the government) content to services such as YouTube, Flickr, Facebook, etc., which can, and often does, spark international outrage and condemnation. The 2009 shooting of Iranian Neda Agha-Soltan is a perfect, but tragic, example of such an event [74]. Her death, captured via cellphone and uploaded to YouTube, went 'viral' on the Internet, shocking the world and putting a face on the pro-democracy, Iranian election protests.

Tragedies of a similar magnitude probably take place everyday, are witnessed and captured by some, but are blocked from the Internet. These tragedies therefore never reach the masses and are highly unlikely to ever force political change. We envision a system that allows citizens with personal mobile devices to share content on a mass scale, independent of government control and censorship of the Internet or cellular communication infrastructure.

1.2.2 Usage assumptions

Based on these two use cases, we envision that CTN nodes will fall into one of two general categories based on the ratio of content producers to consumers in CTN deployments.

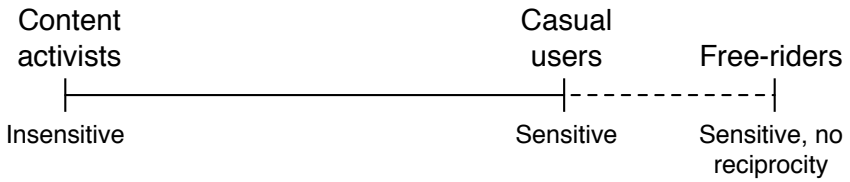


Figure 1.2: Resource-sensitivity spectrum

Highly asymmetric

Highly asymmetric deployments are those with a small number of content producers and a large number of content consumers. We envision that these deployments will arise as a result of activism in the form of citizen journalism. These *content gateways* are assumed to place higher utility on sharing and disseminating information than they do on the resources needed to share the content, and the time and effort to retrieve the content from the Internet or produce it themselves. In practice, we believe that content gateways will be the primary reason for users to participate in a CTN - particularly if the gateways are producing original content that cannot be legally or safely shared openly on the Internet.

Nodes that communicate with, and receive censored content from, a node operating as a content gateway must be trusted by the content gateway (a topic we address in detail in Chapter 4). Although these nodes are free to withhold or further disseminate the content received from a gateway, we believe that nodes that are trusted by a content gateway will themselves operate as sub-content gateways. That is, like content gateways, these nodes place higher utility on sharing and disseminating content, perhaps to a lesser degree, than on the resources needed to share the content. In the absence of a hierarchy of sub-content gateways, content would not propagate beyond the first set of consumers.

Symmetric

We believe that content gateways will be instrumental in getting content from the Internet into a CTN. However, we further assume that the bulk of content will be exchanged in symmetric deployments, where most nodes are sharing and retrieving content from each other. Although we make no assumptions on how content should propagate through a CTN, for content to reach the maximum number of users, it must be shared among what

we generally refer to as *average smartphone users*. These are users that use smartphones as their personal computing device, and are comfortable installing applications on their device.

We expect average smartphone users participating in a CTN to be highly diverse in their goals for the system and their sensitivity to the resources consumed in the pursuit of these goals. Users may be categorized based on their position in a resource-sensitivity spectrum, as illustrated in Figure 1.2 and defined by the following two extremes:

- **Insensitive to resource costs**: In a CTN, users that are insensitive to resource costs and prefer to spread content to as many people as possible are *content activists*. From the perspective of resource allocation, these nodes are considered to be highly altruistic.

  Nodes exhibiting similar characteristics are frequently observed in existing Internet-based peer-to-peer networks [33, 60, 188]. Although their apparent altruism is often rewarded by, for example, an increase in gross data from peers [188] or increased traffic to a website [42]. As we will discuss in Chapter 2, nodes in existing mobile systems generally resemble content activists.

- **Sensitive to resource costs**: At the other end of the resource-sensitivity spectrum are *casual users*. We assume that casual users value the content that they receive through the CTN and allow resources to be consumed; however, we further assume that they value the resources needed to maintain their normal use of the device and expect a reciprocal benefit for any resource expenditure. Casual users will not expend resources to share content with others without an expectation of receiving a commensurate quantity of content in return.

  Nodes exhibiting similar sensitivity to resource expenditure have been observed in Internet-based peer-to-peer systems [28, 133]. In these systems, nodes only contribute as much bandwidth resources as they receive. Though this approach is more "fair", the resulting performance is often far from optimal. In particular, these nodes can create asymmetric links and livelock in a network if reciprocal content cannot be provided to them.

  Users that reside beyond the casual user end of the spectrum, that are not willing to reciprocate the resource expenditure of other nodes to an equal degree, are considered free-riders.

Although there is evidence to support the claim that users may exist at both extremes of the resource-sensitivity spectrum, to the best of our knowledge, no prior work tackles the problem outlined in this thesis (a topic that we further discuss in Chapter 2). In the absence of evidence supporting how future users would be distributed within the spectrum, we engineer for the latter, most rigid usage scenario where users expect a reciprocal benefit from any resource expenditure. Users that are less sensitive, or fully insensitive, to resources may simply ignore, or partially ignore, resource costs and share content without restriction.

A CTN consisting exclusively of resource-conservative, selfish, casual users may not be successful. If demand is not expressed uniformly over the network, nodes that reach the resource allocation limits of other nodes will be starved of subsequent content. Although these nodes may be detrimental to the operation of a CTN, their presence and expectations cannot be ignored. In Section 5.4 we conduct simulations that consist of resource-conservative casual users. In Section 5.5 we further evaluate resource-conservative casual users, however, their resource allocation limits are set sufficiently high that their behaviour is indistinguishable from content activists during the course of the experiment.

1.2.3 A non-networked approach

The goal of this work is not just to build a mobile system, but rather to empower society by resisting, if not defeating, information censorship. There are clearly many ways for censored content to be exchanged. Simple content could simply be printed and widely broadcast as leaflets throughout a region. Richer content could be written to CDs or DVDs and covertly exchanged or anonymously distributed. These techniques are ideal for broadcasting a fixed set of content to a large audience; however, it does not scale to support the dissemination of an arbitrary set of content.

USB keys allow arbitrary content to be stored and exchanged at will. Although the point-to-point 'bandwidth' between two people exchanging USB keys is very high, this approach does not utilize all of the 'connection' opportunities that may exist with other individuals that are, for example, friends-of-friends; increasing the connection opportunities in the system increases its capacity to exchange content. Moreover, the content owned and demanded by the population is likely to be highly diverse [59, 98] and would greatly exceed the capacity of a single USB key. This would require many USB keys and some form of

metadata on the keys governing the state of the shared USB media. We believe that this approach is unrealistic and impractical.

1.3 Threat model

CTNs are designed to broadly circumvent Internet censorship, which may be enforced at various points on the Internet - from the home Internet gateway to tier 1 ISPs. The penalties for violating Internet censorship vary widely; the most significant penalties are those resulting from violating state-mandated censorship [150]. The state is therefore regarded as the primary, most capable attacker.

Limiting access to content is an efficient means of controlling a population and inhibiting social, economic, or political change [150]. Although the act of viewing censored content may be benign, the act of seeking, sharing, or producing content that would be censored, can be regarded as a threat to the state. Websites that advocate democracy, human rights, social justice, etc. are prime examples of this form of content. Individuals violating state-mandated censorship may consequently be considered a threat to the state. Determining the identity of individuals that violate state-mandated censorship is therefore the main asset of interest to attackers.

1.3.1 Capabilities of an attacker

We assume the attackers of a CTN have the following capabilities. In Chapter 8 we will review a CTNs ability to protect the identity of users while subject to each attacker capability.

Internet-monitoring

Attackers have the ability to monitor and control all Internet communications flows both within the country and crossing its borders. Attackers have the ability to perform deep packet inspection on all non-encrypted traffic, and allow or deny any communication flow. Since most states do not operate their own Internet infrastructure, we assume that ISPs operating within state boundaries are fully compliant with the demands of the state.

Although many regimes are subject to trade embargoes that inhibit the sale of computer hardware capable of enforcing Internet censorship, these embargoes can be circumvented by routing the hardware through countries sympathetic to censorship, such as Dubai [187].

Locate end-hosts

Complicit ISPs can further provide an attacker the physical location of an IP address allocated to a subscriber. Through this capability, attackers are able to quickly locate and detain any Internet subscriber that accesses censored content within its borders.

Regulate access to public Internet access points

Attackers have the ability to regulate how public Internet access points, such as those in coffee shops, restaurants, hotels, libraries, Internet cafes, and airports, are accessed. For example, the Chinese government requires that all public Internet access points identify users of their network [18]. This is easily accomplished by requiring users to provide their cell phone number, where a text message can be sent with an access code.

Isolated monitoring

We assume that an attacker has the ability to monitor short-range wireless communication between devices in a specific set of venues. This can be accomplished by a laptop passively collecting packets, or by dedicated monitoring hardware. An attacker can, for example, monitor all communication within popular venues, and subsequently identify communication patterns between users. We further assume that attackers with the capability of monitoring a region are also capable of flooding the spectrum and blocking all communication. We further assume that it is currently not economically feasible for an attacker to monitor all short-range wireless communication throughout a large region.

Infiltrate social networks

We assume that attackers or agents of an attacker, operating under duress or of their own volition, have the ability to infiltrate existing social networks. That is, agents of

an attacker may earn the trust of existing CTN users, and thereby discover information about the network or content within it that would otherwise be private. Attackers that subsequently leave the CTN take all of the discovered information with them.

Physical harm

Unfortunately, the most repressive regimes in the world have the capability to indefinitely detain or execute its citizens [189]. Through this capability, we assume that an attacker that has identified a user in a CTN has the ability to extract all information known to the user. This information, unfortunately, includes the identities of other trusted nodes.

1.4 Contributions

We make three major contributions in this thesis.

- **An architecture for CTNs**

  In Chapter 3 we define a set of design principles for opportunistic communication that are derived from prior work. Following these principles, we present a practical architecture for a CTN in Chapter 4. In addition to satisfying the censorship resistance criteria, our architecture tightly couples network content with node demand and facilitates a pairwise accounting scheme that is not present in existing DTN systems.

  Chapter 5 motivates and defines the *Laissez-faire* framework, a free-rider tolerant approach to CTN routing. We believe that the simplicity of Laissez-faire will serve as the foundation for future DTNs targeted for average smartphone users. We implement the Laissez-faire framework within a prototype CTN system and evaluate Laissez-faire across a cluster of smartphones using a custom-built experimentation platform. Although Laissez-faire does not mandate any specific routing behaviour, we propose several practical strategies for retrieving content fragments. We demonstrate the trade-offs between strategies and the behaviour of Laissez-faire under varying network conditions and in a three-week field trial.

- **SMS-based control channel**

  We characterise the channel characteristics of the Short Message Service (SMS) and design and implement an efficient SMS-based data transport protocol that we call *SMS-TP* in Chapter 6. SMS-TP provides reliable data transport while attempting to minimize message overhead and maximize data throughput of 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). 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.

  This work has been published as [127].

- **Characterization of energy consumption behaviour of smartphone users**

  We study the smartphone usage and energy consumption characteristics of 20,100 BlackBerry smartphone users in Chapter 7. Using this energy traces within this dataset, we build the *Energy Emulation Toolkit*. This toolkit allows developers of energy intensive applications to test their applications' energy consumption behaviour against existing energy traces. Developers may then alter their design or tune algorithm parameters to reduce energy consumption and ensure that their application does not deplete users' batteries.

  We further classify smartphone users into one of three groups according to their unique energy consumption characteristics. We demonstrate that energy level can be predicted to within 7% error within an hour and within 28% error a full day in advance. Using our prediction algorithm, we build the *Energy Management Oracle* library. By querying the library before executing an energy intensive operation, applications can adapt their execution to achieve a near optimal successful execution rate.

  This work has been published as [126, 130].

Chapter 2

Background

This chapter surveys prior work in content sharing between mobile devices. Prior work does not satisfy our design goals. This chapter further surveys prior work related to building a CTN.

2.1 Mobile content sharing

The idea of sharing content between mobile devices is not new. The ability the exchange content directly between two devices safely and reasonably securely is available on nearly every mobile device today. Using Bluetooth for unidirectional content dissemination has been widely discussed [100, 101, 104]. However, this practice requires more manual user intervention and does not scale to allow content to move between potentially non-trusted nodes.

Several BitTorrent clients have been ported to mobile environments that allow mobile devices to download content while within Wi-Fi coverage [10, 39]. This method operates the same as in the desktop setting, with the addition of the resource constraints of the mobile device. We dismiss BitTorrent (in its conventional form) prima facie as a means to share censored content due to its reliance on Internet-based infrastructure.

Decentralized forms of BitTorrent have been explored, where mobile devices exchange content over short-range wireless technologies such as Bluetooth or Wi-Fi, leaving no trace of the data that was exchanged. In BlueTorrent devices exchange fragments of data while connected in a Bluetooth piconet [86]. A similar idea where data fragments are exchanged between cars is presented in CarTorrent [102, 116]. In practice, the set of content that may be available on a mobile device or vehicle is highly diverse [98], so it is unlikely that peers would ever inadvertently retrieve partial files from each other. Instead peers with content would act as distribution nodes, and others would simply retrieve the content that is available via a single hop. Neither BlueTorrent nor CarTorrent satisfy the criteria of censorship resistance.

McNamara et al. examine the problem of maximizing the chances of a successful content download during an opportunistic connection [113]. Transferring content such as songs or movies takes time and consumes valuable resources. To maximize the chance of a successful download, the authors argue that users should identify neighbouring devices that possess desirable content and will remain within range long enough to make a successful

data exchange. They propose a user-centric prediction scheme that collects historical collocation information to determine the best sources of content. This method operates by filtering out neighbouring devices that do not contain content of interest and then ranks contacts in decreasing order according to an estimated remaining collocation time. The device then connects to the top ranked neighbouring device to retrieve a summary of files to download. If the summary contains at least one desirable file, and the *predicted* remaining collocation time is longer than the estimated time to download the file, then the transfer begins. The prediction technique that underlies this method operates by continuously scanning for nearby devices. Devices that are encountered often are classified as 'familiar strangers' [117] and a 'personalized profile' is maintained for that device. Each profile contains the mean collocation time for the device for each time slice within some time period. On subsequent encounters with these familiar devices, the mean collocation time of the current time slice is used to estimate from the remaining collocation time. For devices that are not familiar, i.e. a newly encountered device, a global mean collocation time is used for that time slice to predict the remaining collocation time.

The authors evaluate their technique using a simulation derived from a dataset consisting of over one millions trips on the London Underground[1] by users carrying Bluetooth-enabled devices. They simulate music collections on these mobile devices from a sample of 500,000 users of a popular online "social music" website, Last.fm. The probability that two simulated devices share the same interests in music, and thus make an exchange, is proportional to its popularity within the Last.fm dataset. Their simulation shows that using their prediction technique, which only transfers with contacts with sufficiently long collocation periods, has a success rate of 80%; a 50% improvement over exchanges made with random contacts. Moreover, with perfect knowledge of future collocations, a success rate of 100% is not achievable. They found that with perfect knowledge, roughly 20% of transfers are wasted (incomplete exchanges of files), and that prediction improves efficiency by only 18%. They show that due to the initiation threshold employed by the prediction and perfect knowledge methods, less Bluetooth congestion is present, and subsequently more songs are collected on each simulated device.

McNamara et al. focus solely on single hop exchanges of content, which avoids complex incentive mechanisms: a simple tit-for-tat approach would be sufficient in this scenario.

[1]The subway system uses RFID tags to facilitate easy payment and faster movement through the ticket checkpoints, which allows them to track when a passenger enters and exits the subway system.

However, single hop exchanges could limit the overall quantity of data exchanged between devices and thus underutilize the wireless capacity between devices. It remains an open question if disseminating data as diverse as songs and videos through a single hop is more efficient than multiple hops. Their scheme works under the assumption that people have a high degree of regularity in their movements and that while traveling on city public transport a non-negligible set of passengers will be encountered daily. These assumptions are not always true, and could significantly constrain the effectiveness of applications using this technique.

May, Lenders et. al have created a wireless podcasting system called PodNet [105, 112]. A competing system called HyCast has been also been created by Andronache et al. [16]. In both PodNet and HyCast, users retrieve content from the Internet and advertise their available *channels* and constituent *episodes* to neighbouring devices. In both systems, nodes solicit content of interest during opportunistic wireless connections. As in the previous systems, content is never retrieved on behalf of others. While this approach avoids free-riding, it favours nodes with homogeneous content interests. Helgason et al. extend the PodNet concept with a middleware architecture for a mobile peer-to-peer content distribution [73]. Their architecture allows wireless content dissemination between mobile nodes without the user of infrastructure. In this system content may be exchanged opportunistically when nodes are within communication range. As in PodNet, applications running on top of the middleware may retrieve content feeds from neighbouring devices.

To scale to large numbers of users and accommodate a diverse range of interests [98], we believe that it is necessary for content sharing systems to be able to route content across multiple hops. None of these systems provide that functionality, nor do they satisfy the criteria of censorship resistance.

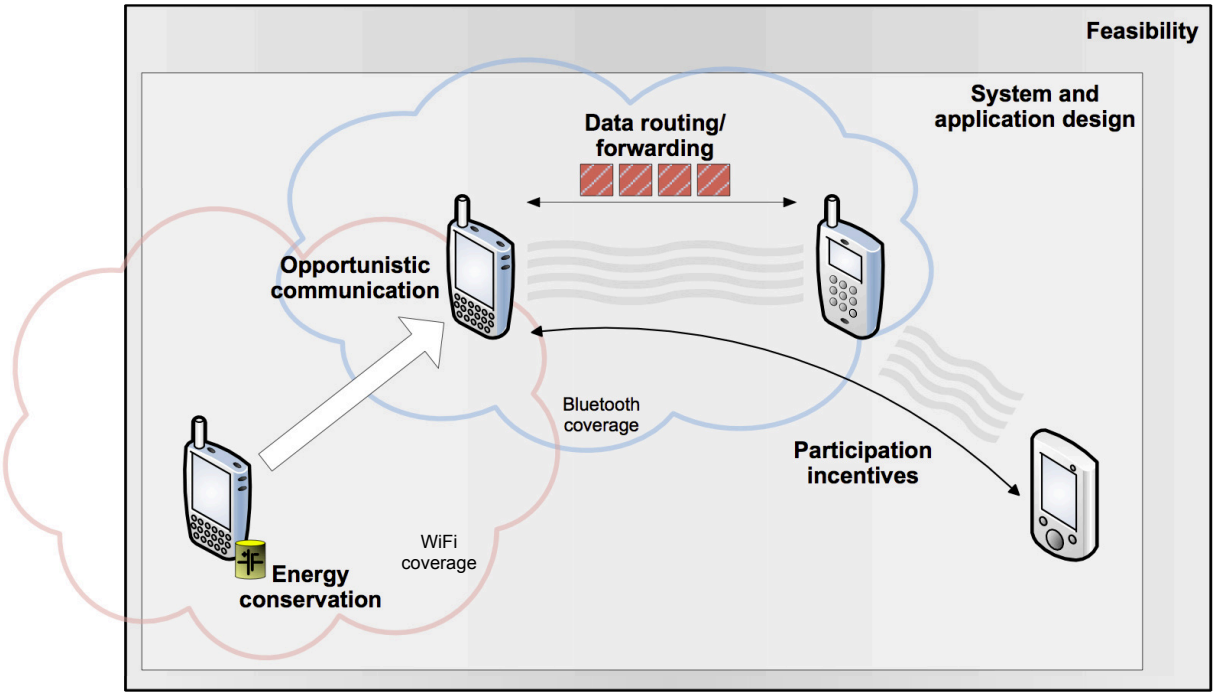


Figure 2.1: Taxonomy of related work: the building blocks of a CTN.

2.2 Building blocks of a CTN

Existing mobile content sharing systems do not satisfy our goals. In the remainder of this chapter we decompose the design of a CTN into several key research domains. These domains are illustrated in Figure 2.1, and will be described throughout the remainder of this chapter. For each domain we conduct a survey of related work relevant to the creation of a CTN.

2.2.1 Feasibility

We begin this survey with a discussion of the feasibility of large-scale DTN-based mobile systems. The work surveyed in this section does not satisfy any of our CTN goals; however, the work does provide context to the problem.

Grossglauser and Tse examine the asymptotic throughput capacity of large mobile

wireless ad hoc networks [62]. Their work shows that direct communication between source devices and destination devices cannot achieve high throughput. Instead, they propose to spread traffic to intermediate relay nodes to exploit the multiuser diversity benefits of having additional routes between a source and destination. Although Grossglauser does not consider end-to-end delay, they prove that mobility significantly increases capacity of ad hoc wireless networks.

Su et al. report on the results of a user study to examine the feasibility of using user mobility and opportunistic contacts to form mobile ad hoc networks [174]. In this work, students carry Bluetooth-enabled devices for several weeks. Their work shows that networks can be successfully formed under the conditions of user mobility, and that there is potential for exploiting the regularity in contact behaviour to improve message transport between devices. One key observation from their experiment was that less than 18% of messages were delivered via a single hop. The remainder of messages was delivered faster using some multi-hop path. This seminal experimental work provides insights into the capabilities of much larger networks.

Kathiravelu and Pears further explore the capacity of 'opportunistic[2] networks' [91]. In this work they present a simulation of 50 to 200 Bluetooth-enabled devices operating in a simulated airport environment. The simulation and results presented in this paper are weak primarily due to their use of a random waypoint model, which has been proven to poorly model human mobility [153]. However, this paper draws our attention to several factors that must be considered in evaluating the capacity of opportunistic networks. First, Bluetooth devices must first be discovered before they can communicate with each other; being within range of another Bluetooth device is not a sufficient condition for an opportunistic connection to take place. These issues are addressed in Reference [119], and will be further discuss in Section 2.2.2. Second, mobile Bluetooth-enabled devices such as cell phones and smartphones are highly resource-constrained, which often inhibits their ability to transmit data at the full rate supported by the wireless chipset. A discussion of how resource constraints affect opportunistic communication can be found in our previous work [128, 129]. Techniques for adapting to the multitude of resource constraints

[2]'Opportunistic' is an overloaded term in the networking research community. It is most commonly used to refer to networks where in the absence of network infrastructure, content of interest is transferred between mobile devices using connection opportunities that arise when devices are within wireless range of each other.

common to mobile devices remain an open problem. Third, our prior work has shown that the process of selecting data to forward to other nodes during an opportunistic connection further reduces the amount of data transferred and reduces the capacity of these networks [143, 144].

2.2.2 Opportunistic communication

Devices participating in CTN must be able to communicate robustly despite potentially short and infrequent connection opportunities. This section explores work in this crucial problem domain.

Liberatore et al. explore the problem of maximizing transfer opportunities between mobile devices using Bluetooth [107]. As one of the most inexpensive, energy-efficient and widely deployed peer-to-peer capable radios, Bluetooth is well suited for use in a DTN. Unfortunately, Bluetooth's half-duplex process of neighbour discovery can take tens of seconds to complete between two mutually undiscovered radios, which is often longer than two devices are within proximity. To solve this problem, the authors introduce a second Bluetooth radio to provide a full-duplex inquiry channel. The use of two radios would allow a device to both inquire for neighbouring devices, and listen for incoming connections. Through analysis and simulation[3] they show that using two radios reduces the time until nodes discover each other by 12 to 25%. This reduction translates into an estimated 170 to 440% increase in the number of discovered transfer opportunities. They also showed that dual radios are more energy efficient, spending 7 to 27% less energy on average per second of data transferred. While their work yields significant improvements in both connection frequency and duration, the addition of a second radio in a mobile device is impractical for the foreseeable future.

Jung et al. present BlueTorrent, which allows collocated Bluetooth devices to form a piconet to disseminate data between them [86]. A key challenge inhibiting BlueTorrent, and systems that use Bluetooth, is efficient peer discovery. Unlike the previous paper, this work assumes that only one Bluetooth radio is available in each device. Recall that for two nodes to connect using Bluetooth, one node should be in the inquiry state and the

[3]The simulation uses a mobility model based on random, predetermined paths, which has been shown to poorly represent human mobility [153].

other in the inquiry scan state. However, since Bluetooth users are typically moving, their roles as inquirers (masters) or inquirees (slaves) cannot be predetermined. These roles must therefore be randomly alternated. Bluetooth supports such a random role selection through the periodic inquiry mode. In this work, the authors approach the neighbour detection problem by analyzing the Bluetooth periodic inquiry mode. Through their analysis and simulation, they show that the timings used within the periodic inquiry mode are sub-optimal, the inquiry scan interval is a critical factor, and that reducing the interval to 0.32 seconds minimizes discovery latency at 2.53 seconds.

Unfortunately, the work by Liberatore and Jung require modification to either the physical hardware or Bluetooth stack, which is currently not feasible in most mobile development platforms.

The problem of maximizing communication during an opportunistic connection is exasperated by increased speed and shortened connection periods that are introduced when users are within vehicles. Ott and Kutscher investigate the practicality and challenges in providing network connectivity and Internet access to mobile users in vehicles [132]. They evaluate the performance of 802.11b under several scenarios using both TCP and UDP to transfer data from a mobile client (laptop) to a server behind a roadside access point. They show that the total amount of data that can be transferred decreased proportionally to the vehicle's speed: approximately 8.8 MB at 80 kph, 7.8 MB at 120 kph, and 3.7 MB at 180 kph for UDP traffic sent from the client to server. They also observe an asymmetric relationship between UDP traffic. UDP data sent from the server had a minimum loss rate of nearly 60%, with a corresponding decrease in throughput. Using TCP they observe that a total of 6, 5, and 1.5 MB of data can be transferred at 80, 120, and 180 kph respectively; nearly a factor of two improvement over UDP transmissions from server to client. Moreover, they observed similar that throughput using TCP was similar in both directions.

Hadaller et al. further examine the behaviour of 802.11g under drive-by scenarios [70]. Through experimentation in real-life drive-by scenarios, Hadaller shows that on average, current protocols only achieve 50% of the overall throughput possible in a drive-by scenario. They further show that "high packet losses early in a connection are responsible for the loss nearly 25% of overall throughput, 15% of the time."

Although the measurement studies by Ott and Hadaller are not specifically relevant to the creation of a CTN or provide solutions to any of the goals, their work provides insight

into the service quality that mobile users can expect when using their device to interact with roadside Wi-Fi infrastructure (if desired).

2.2.3 Participation incentives

Device participation in a DTN requires the device owners to invest energy, storage, computation, and possibly money. This section outlines techniques used to incentivize participation in existing systems that exchange data between mobile devices.

Ananthanarayanan et al. present a system for collaborative downloading that uses both the WLAN and the WWAN in combination, in an attempt to "bridge the range-speed dichotomy" [13]. In *COMBINE*, devices in close proximity use the high-speed WLAN interfaces to discover each other, form a collaboration group, and stripe static HTTP traffic across their WWAN links. This allows any one device to increase its effective WWAN download speed, in a manner that is transparent to the user. A significant contribution of this work is COMBINE's approach to buying and selling WWAN bandwidth, and thus providing incentive participation.

In COMBINE, each device determines the energy cost associated with downloading over a WWAN interface. This estimation, along with the monetary cost of downloading data, is used to determine the price of using the device's WWAN bandwidth. Devices running this service periodically announce their presence and their bid, which contains the collaborator's price and an estimation of the WWAN download speed that it is able to offer. When a device requires bandwidth neighbouring devices, it notifies its neighbours of the website URL that it wants to download, and waits for a small duration. If a neighbouring device has the website in cache, it can then unicast a response to the initiator indicating that it has the file. Otherwise, the initiator forms a download group through one of two approaches. The first method narrows the set of neighbouring devices to those whose bid is less than the cost per byte that the initiator is willing to pay. The initiator then selects the group members in order of decreasing bandwidth. In the second method the initiator considers all devices, including those with bids higher than its cost per byte threshold. The initiator selects group members according to an optimization problem that minimizes the total time taken to download the desired file while maintaining a maximum cost C.

Once a collaboration group has been formed, the initiator distributes work items using one of two strategies. In the first, the initiating device gets the total size of the file to

be downloaded and forms a work queue that specifies fixed equal sized byte ranges of the file to download. Collaborating devices then pick up data from the work queue, download the data specified in the work item, and return the data-initiating device. In the second strategy the initiator divides the file of size F into fixed-size partitions of size p bytes each and apportions to each partition a cost budget of $(C/F)p$. The initiator then allocates work items to collaborators explicitly to ensure that the budget is satisfied. Payment to collaborating devices is achieved using a system of signed IOUs that are issued by the initiator after each work item is completed. Collaborating devices store the IOUs and can redeem them at a later time by contacting a centralized, trusted accounting server. The accounting server maintains record of redeemed IOUs to prevent duplicate redemption.

While the IOU approach to enabling participating incentives satisfies the requirements of COMBINE, it does not solve the incentive problem underlying communication in a DTN. In COMBINE, the reward of an IOU is tightly coupled with the act of downloading data; an IOU is not issued until the participating node has delivered an intact work unit. However, in a DTN, participating devices must store, carry, and forward data independent of the source or destination. Moreover, data relayed to one device may never reach another device or may be replicated to many devices. Due to the tight coupling of actions and rewards, COMBINE's approach to providing incentives does not satisfy our CTN design goals. Further, the use of a centralized accounting server further violates the unlinkability constraint and could lead to the loss of anonymity for users of the system.

We further explore strategies for incentivizing participation in DTNs in Section 2.2.6.

2.2.4 Energy conservation

Energy conservation is a fundamental concern in the design of any mobile system. Systems that consume energy too aggressively will drain the host device's finite energy resources, and are likely to be deleted from the host device.

The wireless subsystem consumes a significant portion of total power consumption [11, 139]. A primary source of wasted power consumption in mobile ad hoc networks is scanning for neighbouring devices; a scan that yields no neighbouring devices provides no real utility and consumes limited energy. Galluccio et al. have proposed an analytical framework for studying the trade-off between energy efficiency and time for neighbour discovery

process in ad hoc networks [56]. Sedov et al. discuss an approach for making the Bluetooth service discovery protocol more energy efficient [162]. Several authors propose and analyze symmetric neighbour discovery schemes [32, 156, 169]. However, these techniques all require making changes to the Bluetooth software stack to adjust the timings within the inquiry and scanning processes. In practice, altering the established Bluetooth protocol is not feasible. Drula et al. consider the problem of performing energy efficient Bluetooth neighbour discovery without modifying the existing Bluetooth software stack [46, 47]. In this work they present two algorithms that select the method of neighbour discovery: either low-power slow discovery or high-power fast discovery. Their first algorithm uses the level of recent activity to determine scanning mode: recent device detection is a strong indicator that a subsequent scan will discover devices. The second scheme relies on location sensing and the location of previous contacts to determine the scanning mode. For example, if few neighbours are detected in the vicinity of location A and many neighbours are detected in the vicinity of location B, then a device should operate in low-power mode at location A and high-power mode at location B. This work further shows that the mean discovery time can be as low as 0.2 seconds, contradicting prior work by an order of magnitude [23, 159, 200]. Although both techniques appear to be relevant to the implementation of a CTN, real-life evaluation in a diverse range of usage scenarios is needed to realize the true benefit of each scheme.

A process similar to the efficient detection of neighbouring devices is the process of detecting neighbouring Wi-Fi access points. Nicholson et al. present BreadCrumbs [123]. BreadCrumbs is a network middleware that exploits the fact that users of mobile devices are typically creatures of habit, whose mobility is limited to few fixed regions. BreadCrumbs tracks the movement of the device using beacons from existing Wi-Fi AP[4] and GSM cell towers and builds a mobility model of that user (similar to the technique used in Reference [111]). This model allows the middleware to predict both future network connectivity and level of throughput that applications can expect. This has two significant benefits to the creation of a CTN. Although BreadCrumbs uses Wi-Fi APs for location sensing, coarse-grained positioning using purely cell tower triangulation is probably sufficient for many applications. Using even a rough prediction would allow a device to save energy by disabling its Wi-Fi radio an AP is predicted to be within range. Predicting the time until Wi-Fi availability and future throughput can further improve the service

[4]Using an AP triangulation method first presented by LaMarca et al. in Place Lab [97].

quality of delay tolerant applications that utilize multiple network interfaces. If an application predicts (with a high probability) that sufficient Wi-Fi service will not be available within some deadline, then the application can avoid unnecessary delays and immediately transmit data over an alternate interface.

Techniques like BreadCrumbs that rely on existing knowledge of Wi-Fi access points perform poorly when users are highly mobile. When a mobile device has data to send, it is often necessary to scan for access points. Work by Falaki reveals that by using a small cache of Wi-Fi access points and the GSM cell tower IDs within their range, a 46% reduction in scans can achieved over static scanning techniques [53].

CoolSpots by Pering et al. examine how multiple network interfaces can be used to minimize energy consumption [138]. CoolSpots is a hybrid Wi-Fi and Bluetooth system that provides improved communication capabilities to devices that are within a 'CoolSpot-enabled' region. The system uses Bluetooth-enabled access points, which allows mobile devices to take advantage of the diversities between the two radio technologies. Low bandwidth applications can connect to access points over Bluetooth, and when higher bandwidth and/or an increased transmission range is needed, the device can switch to Wi-Fi. Underlying CoolSpot's system for switching between network interfaces is a policy language that allows applications to specify their bandwidth, power, and range requirements. Their results indicate that more than a 50% reduction in energy consumption is possible, with an associated increase in the effective battery lifetime. Although we believe that the dependency on CoolSpot-enabled infrastructure makes this work difficult to deploy, the insights gained by their energy management policy could be applied to device-to-device communication and have applications in our work.

Banerjee et al. present a ten person study that measures how users of mobile devices consume battery [19, 147]. Although their study is preliminary, the paper highlights the diversity of energy consumption habits among users. Falaki et al. expand upon this work by measuring the energy consumption and application usage behaviour of 33 Android and 222 Windows Mobile devices [52]. They subsequently build tools to visualize and identify sources of energy consumption on mobile devices [51].

Although a great deal of work has been done to minimize energy consumption by wireless interfaces and categorize sources of energy consumption, no work has been done how energy-intensive applications themselves should respond to fluctuating energy reserves. As

a potentially energy demanding application, the operation of a CTN can be significantly affected by users' energy consumption habits. For example, if a user charges her device daily, then her device may scan for neighbouring devices and APs more frequently. Conversely, a device that is charged shortly before losing power has little capacity for extraneous wireless communication. We believe that it is essential for energy-intensive mobile applications to adapt their execution to changing levels of energy reserves. We will revisit this problem in Chapter 7.

2.2.5 System and application design

Several related projects have explored the use of opportunistic communication to either disseminate data between mobile devices or to communicate with servers on the Internet. The most relevant work is within the context of the Haggle Project [75, 76, 161], the Opportunistic Connection Management Protocol (OCMP) [164], and the MobiClique mobile social networking system [143, 144].

Haggle is a mobile application middleware that is designed to separate application logic from transport bindings so that applications can be agnostic to the underlying network interface. Applications delegate the task of handling and communicating data to Haggle, which in turn adapts to the current network environment using the 'best' available forms of connectivity. For example, the Haggle middleware is designed to allow applications, such as email, to transport application data either between devices using opportunistic communication or by using existing infrastructure, such as SMTP servers on the Internet. However, most of the research within the Haggle Project has focused solely on infrastructure-less communication, where mobile devices communicate with each other to disseminate data over a DTN.

Seth et al. define the Opportunistic Connection Management Protocol (OCMP) [164]. OCMP allows applications to communicate opportunistically over multiple heterogeneous wireless networks. Like Haggle, OCMP abstracts network connectivity from applications, which forces applications written for OCMP to be delay tolerant. Internally, OCMP also views network connectivity as a schedulable resource, which can consume network time, battery power, monetary costs, etc. However, unlike Haggle that relies solely on Wi-Fi, network scheduling is controllable through application specific policy. A similar technique

can be found in the Horde middleware [146], which allows applications to specify quality-of-service objectives when transmitting data over a heterogeneous set of varying wireless network channels.

OCMP further differs from Haggle by its integral use of infrastructure[5] on the Internet. Communicating over multiple network interfaces with intermittent connectivity requires that OCMP first fragment data on the mobile device. Fragments are transmitted independently over available network interfaces to a centralized server, or 'proxy' on the Internet. At the proxy, data fragments are reassembled into their original data and forwarded to legacy servers. For example, an email sent from a mobile device containing a large photo attachment is fragmented on the device, transmitted over one or more wireless networks to the proxy. At the proxy the email is reassembled and forwarded to a SMTP server for delivery to the recipient.

The third system, MobiClique by Pietiläinen, Oliver et al. [143, 144], is designed to provide a system of participation incentives in a DTN. MobiClique builds on the Haggle philosophy of disseminating content purely between mobile devices and operates in a fully decentralized manner; however, it depends on an existing online social network, such as Facebook, to bootstrap its operation. The design of MobiClique was motivated by the assumption that people will forward data on behalf of others if they are friends, friends-of-friends, or share similar interests; online social networks are the glue that represents these criteria.

In MobiClique, each participating device stores the identity of its user and the identities of all of their friends (as obtained from the online social network). As users and their devices roam around and detect other devices, they each exchange their identity and the identities of their friends. Using this information, each device forms a directed graph that is an aggregate representation of social relationships. MobiClique then uses the social network graph to make forwarding decisions. During an opportunistic connection, data is forwarded only if the neighbouring device shares some social connection with the recipient. The result of our experimentation and social network analysis are presented in References [118, 143]. MobiClique's key contribution was the lessons learned from its

[5] 'Infrastructure' is loosely defined in the literature to mean both communication infrastructure like cellular and Wi-Fi networks, and computing infrastructure such as servers on the Internet. In this proposal, we assume that communication infrastructure is ubiquitous and that 'infrastructure' refers to computational capacity.

failures. As the first working prototype of its kind[6], MobiClique revealed many system constraints that inhibit the effectiveness and reliability of systems that use opportunistic communication. For example, during an opportunistic connection, each device must first select a subset of data to forward, which can be both a computational and I/O intensive operation, then load the data from slow persistent storage into memory, and transfer the data over the network interface. Depending on the device, this operation can trigger either virtual memory page flushes or garbage collection, which further constraints data transfer. The initial MobiClique prototype also used Bluetooth to discover and communicate with neighbouring devices. Although Bluetooth reduces the energy cost needed to discover neighbouring devices, it does not maximize the amount of data that could be transferred between two neighbouring devices. Switching to Wi-Fi, for example, would significantly increase the amount of data exchanged between two neighbouring devices. This observation motivates the need for multiple network interfaces; however, in the presence of human mobility, variable setup times, and different energy requirements, the problem of how and when to use multiple network interfaces efficiently remains an open problem.

None of the three aforementioned systems satisfy our goals. OCMP's use of a centralized proxy server violates the unlinkability constraint. Like BitTorrent, users of the OCMP service could be identified purely through their communication with the proxy server. MobiClique and Haggle are both robust to attack and scale; however, they do not provide security or unlinkability. Any user running the software may interact with any other node. It is therefore not safe to share censored content in either system.

Several other systems have been built based on a message ferrying approach that utilizes mobile nodes to store, carry, and forward data to other nodes in the network. The KioskNet [66, 67] project builds on previous DakNet work by Pentland et al. [137]. In DakNet vehicles, such as public buses, are equipped with access points. As the vehicles drive through rural communities, data is uploaded from village kiosks to the mobile access point. The data 'bundles'[7] are then transported to an urban center where Internet access is available. The data is then transferred over the Internet to a proxy and forwarded to legacy servers. This technique has been termed 'mechanical backhaul'. KioskNet differentiates itself from DakNet by supporting all forms of available connectivity, including

[6]At the time, the prototype Haggle implementation was in Java and only ran on a high performance laptop.

[7]Packets transported over a DTN are commonly referred to as bundles [54].

long-range wireless, SMS [127] (and Chapter 6), Wi-Fi in conjunction with mechanical backhaul, cellular data services, and USB key-based sneakernets.

KioskNet's security architecture allows disconnected nodes to securely communicate with all other disconnected nodes. Secure communication is achieved by storing the public keys of users in a central trusted database on the Internet. By periodically disseminating the database to all rural kiosks, applications may encrypt data destined for any other user in the network using the recipient's public key. Similarly, rural kiosks can communication securely with the proxy by encrypting data using the proxy's public key[8]. KioskNet security architecture therefore enables e-commerce and other privacy sensitive applications to be deployed in disconnected regions [185] . While KioskNet's approach to providing secure communication to disconnected nodes is both simple and practical, it does not satisfy the criteria of censorship resistance due to the need to trust the service provider. KioskNet is thus not suitable for widespread sharing of censored content.

2.2.6 Data routing and forwarding

Borrel et al. provide a classification framework that formally describes the relationship between DTNs and mobile ad hoc networks and how they change over time [24]. They argue that MANETs are simply a special case of DTNs, and that DTNs are a special case of disconnected networks where space-time paths do not exist. This classification is important to understanding where the proposed research fits into the body of existing 'mobile' routing research. Using terminology presented in Borrel's paper, this survey will focus purely on routing in assisted-DTNs, where there may not exist a space-time path between all participating node pairs. However, as evidenced by an existing survey [83], there is an extensive body of work even in this sub-domain. We therefore focus on the subset of work that could be relevant to the creation of a CTN.

Jain et al. formulate the DTN routing problem, where messages are moved from end-to-end across a time-varying connectivity graph [80]. In these networks, data is stored at intermediate nodes until a connection, or edge, to the next node in a routing path can be made. The main contribution of this work is their outline of several different routing algorithms and framework for evaluating each algorithm. They distinguish algorithms based on

[8]Or more precisely, using a symmetric key encrypted using the proxy's public key.

the amount of knowledge that an algorithm has about the contact patterns of nodes, buffer capacities at nodes, and traffic patterns. *Zero knowledge* algorithms that operate independently of past or future information performs poorly in non-trivial topologies because the chosen next hop is essentially random and exchanges with a neighbouring node may not transport data closer to its destination. *Partial knowledge* algorithms utilize knowledge about both contact data and/or buffer capacities. These algorithms are based upon assigning costs to edges in the time-varying connectivity graph and finding the cheapest path using a modified version of Dijkstra's Algorithm. The first algorithm seeks to minimize the expected delay by assigning edges the sum of the average waiting time, propagation delay, and transmission delay. This algorithm is the most practical as it relies solely on average contact statistics, which can be approximately by each node over time. The second algorithm uses knowledge of expected contact patterns to determine routes that result in the earliest delivery of a message. Paths are computed at the source node without considering the availability of storage at intermediate nodes, which in practice would lead to message drops when buffers overflow. The third algorithm accounts for the buffer sizes on all edges leaving the current node, which the authors show significantly reduces delay under heavy workloads. In this algorithm, routes are recalculated at every hop to take into account buffer sizes at all edges in the path. The fourth algorithm uses knowledge of buffer capacities to determine the instantaneous buffer sizes across the entire topology at any point in time. Like the previous two algorithms, this information is combined with knowledge of contact patterns to determine routes that result in the earliest delivery. Like the second algorithm, routes are predetermined at the source since buffer capacities are known a priori. Interestingly, they show that this algorithm yields negligible improvements over the third algorithm that does not have global knowledge of buffer capacities. The third class of routing algorithm presented in this paper utilizes *complete knowledge* of contact patterns, buffer sizes, and node movement to find the optimal path. The authors present a detailed linear programming technique for finding optimal routes to minimize average delay in the network. The authors show through simulation that the performance of 'smarter' algorithms that take consider contact patterns and time-varying link conditions perform best, both in terms of delay and delivery ratio. In general, the performance of an algorithm is proportional to the amount of knowledge that it has. In practice, devices operating in a DTN are not privy to future knowledge; however, as alluded to by the authors, the use of a low-delay control channel to provide global state could simplify some the routing problem.

Jones et al. present an alternative approach to routing in a DTN that does not rely on future connectivity knowledge [84]. Their technique, *Minimum Estimated Expected Delay* (MEED), relies solely on observed information about the network and follows the same approach previously used by Jain et al. to minimize end-to-end delay. Instead of computing the expected waiting time using the future contact schedule, Jones's technique records the observed contact history at each node and floods the 'link state' information to the other nodes in the network. Routing decisions are then made at each node to ensure that data is routed according to the most recent information. Through simulation they show that MEED achieves 96% of the delivery ratio achieved when using epidemic routing and that MEED approaches the performance of Jain's algorithms that require future knowledge of the network topology.

Leguay et al. present the MobySpace routing scheme [103]. In MobySpace, each device records its mobility model as the probability in being in each of n fixed known locations, or *MobyPoints*. The mobility pattern is then flooded to all other nodes in the network. Routing decisions using MobySpace are then made under the assumption that a node is a good candidate for taking custody of a bundle if it has a mobility pattern that is similar to that of the bundle's destination. Routing is then accomplished by forwarding bundles toward nodes that have mobility patterns that are increasingly similar to the mobility pattern of the destination. This technique is unique in that it tightly couples human mobility with routing, which we believe is essential to routing in a DTN. Unfortunately, this technique is only suitable for small-scale deployments, such as within a school, where most of the nodes can be represented by a reasonable number of MobyPoints. As the radius, r, of the networked region expands, the number of location points would likely increase by $O(r^2)$, which makes MobySpace impractical for large-scale deployments.

Chaintreau et al. assess the impact of human mobility on the design of opportunistic forwarding algorithms [29]. They observe that the distribution of the inter-contact time, that is the time gap separating two contacts of the same pair of devices, exhibits a heavy tail such as one of a power law, over a large range of value. These results were also observed in Reference [143]. Based on this observation, they prove that forwarding algorithms based solely on the destination address/identity may deliver data with a bounded expected delay in the case of light tailed inter-contact times, as well as when the mobility of devices creates inter-contact times that follow a power law with coefficient greater than 1. However, they also prove that these simple algorithms have an infinite expected delay when the inter-

contact times follow a power law with coefficient smaller than 1.

Lindgren et al. propose PRoPHET, a Probabilistic Routing Protocol using History of Encounters and Transitivity [108]. PRoPHET is based upon a probabilistic metric called the "delivery predictability" at every node a for each known destination b. The delivery predictability is used to represent how likely it is that this node will be able to deliver a message to that destination. When two nodes meet, they exchange summary vectors, and also a delivery predictability vector containing the delivery predictability information for destinations known by the nodes. Each node subsequently updates its own internal delivery predictability vector. Once delivery predictability vectors are exchanged, the updated vectors are used to decide which messages to request from the other node. The technique used in PRoPHET is not sufficient for use in a CTN for two major reasons. First, there are many factors that in practice could impact delivery probability. For example, the size of a message and its corresponding resource cost could significantly impact delivery probability. Users may be willing and able to transport only censored content that is small in size. Second, their technique for ranking nodes based on delivery probability is impractical under our design assumptions. Nodes that are often encountered have a higher delivery probability; however, free-riding nodes can falsify this state and report that all other nodes have low or zero delivery probability. PRoPHET's dependency on falsifiable link-state makes it unsuitable for use a CTN. This critical limitation will become a common theme throughout this section.

MV Routing by Burns et al. learns structure in the movement patterns of nodes [26]. MV maintains information about *meetings* between nodes and their *visits* to locations and uses this information for routing and buffer allocation. In MV, when two nodes encounter each other, each node discloses the list of messages that it carries along with a likelihood of delivering each message. Each node sorts the unioned list of messages by delivery likelihood and deletes the messages that the other node has a higher likelihood of delivering. Each node then selects the top n messages and requests the messages that are not already stored. The likelihood of delivery of a message is computed by each node based on the probability of visiting the message's destination region. MV assumes that the probability of visiting a region in the future is strongly correlated with the node's history of visiting a region. Free-riding nodes could easily report low visitation probability, and thus legitimately delete all unwanted messages or simply delete everything and never report holding any messages.

Hui et al. create the BUBBLE forwarding algorithm that exploits observations that

human interaction is heterogeneous both in terms of popular individuals and groups or communities [77, 197]. The authors identify through examination of existing mobility traces, that within a community of cooperating individuals, some people are more popular, and interact with more people than others. These individuals have high centrality and are 'hubs'. BUBBLE utilizes community information and correlated interaction to select forwarding paths. Forwarding between nodes under BUBBLE is dependent on a global ranking of all nodes in the system and a local ranking within a community. If a node has a message destined for another node, this node first bubbles the message up a hierarchical ranking tree using the global ranking, until it reaches a node that is in the same community as the destination node. Once a message has reached the destination community, the message continues to bubble through a local ranking tree until the destination is reached or the message expires. The authors show that BUBBLE outperforms PRoPHET; however, its dependence on a global ranking of nodes and community structure makes it unsuitable for use in a CTN.

Zhao and Ammar introduce the *Message Ferrying* routing scheme [206]. The message ferrying routing scheme differs from previously discussed routing approaches primarily through its use of a low-bandwidth, long-range control channel. In the *node-initiated* variant of this scheme, message ferries move around in a deployed area according to known routes and communicate with other nodes they meet. With knowledge of the ferry routes, other nodes can adapt their trajectories to meet the ferries and transmit or receive messages. Zhao's definition of 'message ferry' differs from its use in other DTN-based systems [67, 137] that loosely define message ferrying as the act of storing, carrying, and forwarding data; independent of the routing protocol used. This scheme further differs from previous schemes in its non-random proactive movement of nodes. For example, both KioskNet and DakNet, use epidemic routing [186] to disseminate data from the Internet to rural kiosks, which may lead to significant and even unacceptable transmission delays and low throughput. As a proactive routing scheme, Message Ferrying proposes that mobile nodes actively modify their trajectories in order to transmit messages as soon as possible. The key contribution of this paper is the ferry route-finding algorithm that minimizes delay while satisfies minimum bandwidth requirements. The authors divide the route-finding problem into two sub-problems. The first problem seeks to find a route that minimizes the average delay for a random set of expected traffic without considering the end-to-end bandwidth requirements. They show that this problem reduces to the (NP hard) Euclidean

Traveling Salesman Problem (TSP), and approximate it using existing TSP solutions. The second sub-problem extends the route generated in the first sub-problem, if necessary, to meet the bandwidth requirements. Through simulation they show that with proactive movement of both the ferries and the nodes, the message ferrying scheme, in theory, provides regular connectivity in an otherwise disconnected ad hoc network.

Zhao et al. introduce the *ferry initiated* message ferrying (FIMF) routing scheme [205]. Under FIMF, when a node wants to send packets to other nodes or receive packets, it generates a service request and transmits it to a chosen ferry using a low bandwidth, long-range radio. When a ferry receives a service request, it alters its trajectory to rendezvous with the node. While navigating to the node, the ferry periodically broadcasts its position to nodes. Similarly, if nodes move from the location where they initiated the service request, they must broadcast location updates for the ferry. The ferry then recalculates its route to rendezvous with the node at its new location. Data exchange takes place when the ferry reaches the node. After servicing the node, the ferry then resumes a known default route through the deployment region while continuing to broadcast its location and listen for service requests. If additional service requests are received, the ferry services them using one of two approaches. The first approach services the nearest neighbouring node. The second approach uses a traffic-aware heuristic that considers both the location of the node and its message drop information. The message drop information is collected from each visited node. It contains the message generation rate and the cumulative number of messages. Given that buffer sizes are assumed to be constant, the ferry then calculates the message drop rate at each node.

Zhao and Ammar's papers are among the first work to propose proactive movement of mobile devices to forward data. Proactive movement of nodes is further studied by Burns et al. [26]. They discusses and evaluate the impact of autonomous nodes moving around within a DTN and altering their mobility in response to changing network capacity and demand. Although there are few circumstances today where mobile users would alter their mobility to satisfy the data forwarding needs of their device, we believe that the idea has merit and could be used in a CTN. For example, if a user was somehow informed that some content was available at a nearby location, she may be inclined to alter her plan to retrieve the data. To the best of our knowledge, no further work has considered a *user*-driven approach to routing content within a DTN.

An alternative approach to disseminating content is to simply flood the content through-

out the network. Vahdat et al. refer to this as Epidemic routing [171, 186]. Unfettered epidemic routing is infeasible in a DTN if the aggregate size of content within the network is greater than the storage capacity on any node. Nodes must be highly selective in the data they retrieve to satisfy storage constraints. One approach to this problem is to produce an unlimited number of copies of messages, but restrict the message to travel a maximum of number of hops from the source [186]. Similarly, Reference [171], restricts the number of message copies, which limits both the depth and breadth of the resulting flood tree. However, under our design assumptions it is not possible for individual nodes to dictate operational behaviour to other nodes in the network. Nodes are assumed to operate independently and rationally, and thus allocate their limited resources as they deem to be in their best interests. Epidemic routing is suitable for disseminating small amounts of data, such as control information, where the aggregate size of the data is assumed to be small. We will revisit the limitations of epidemic routing in Chapter 5.

Epidemic routing is not suitable for the reasons previously discussed. Prioritized Epidemic Routing Protocol (PREP) defines a *average availability* metric may be a useful metric for ranking nodes [148]. PREP operates by measuring the average fraction of time that a link will be available for use. The value is calculated based on the amount of time that the link was available over some duration. The protocol then advertises this quantity to all other nodes in the network through standard epidemic routing (*a topology sync*). This best effort topology awareness mechanism is subsequently used to compute routing costs. The shortest path is then computed using Dijkstra. PREP prioritizes messages as follows: messages with an expected hop count exceeding some threshold are assigned a priority based on their shortest path length. The higher the path, the lower the priority. PREP then disseminates messages with the lowest priority. PREP keeps storage and bandwidth utilization maximized and drops data only when necessary. When resources are consumed, messages are deleted in order of highest priority. Unfortunately, in practice, malicious nodes could share invalid average availability vectors to create artificially long paths through themselves. There is also no mechanism to prevent a PREP node from deleting all of its messages. In addition to sharing invalid availability vectors, malicious nodes are free to drop all data without consequence.

Gradient routing protocols operate by computing a delivery metric for all destinations [172]. Messages are forwarded when a node is encountered with a higher delivery metric. ZebraNet is system that uses gradient routing protocol to flood messages towards

fixed base stations (gateways) [85]. Ignoring the fact that CTNs do not have fixed base stations, grading routing protocols would not work because the aggregate quantity of data in the network could easily exceed node storage capacities. Moreover, these techniques require more knowledge of the system than can be guaranteed to be available in a CTN.

Costa et al. propose SocialCast, a publish-subscribe based routing protocol that exploits predictions based on metrics of social interaction (i.e. patterns of movements among communities) [41]. A basic assumption underlying SocialCast is that nodes that have the same interests spend time collocated. Routing in SocialCast consists of three phases: interest dissemination, carrier selection, and message dissemination. During the Interest Dissemination phase, each node broadcasts a control message containing its interest strings and utility vector to its direct neighbours. The utility vector contains a list of value derived from collocation with other nodes. During the Carrier Selection phase, the utility vector of the local node is recomputed for all interests. This utility of each interest is compared against the utility vectors received by neighbours. If any neighbour has a higher utility, then they are a better carrier for the data, and the content is transferred to them in the next phase. Otherwise, the local node is still the best carrier for messages tagged with the interest string. During the Message Dissemination phase, the content of the buffer is re-evaluated against the new subscriptions and utilities, and messages are forwarded to the interested nodes. A copy of messages matching an interest is immediately sent to all neighbours whose subscriptions contain the interest. SocialCast is not suitable for use in a CTN for two reasons. The first obvious, common problem is that malicious nodes to exclude themselves as carriers can easily falsify utility vectors. Second, SocialCast's expression of interests is imprecise and in practice would lead to efficient network utilization. We will revisit this topic in Chapter 5.

As we have seen, none of the previously discussed routing protocols are suitable for use in a CTN. In addition the use of falsifiable link-state, none of the previously described routing techniques are incentive compatible. For DTNs to be used on personal mobile devices, there must be some mechanism to incentivize participation. We now consider recent work that aims to incentivize participation in a DTN.

Participation incentives in DTN routing

The lack of participation incentives has been a major problem ignored by existing DTN systems designed for personal mobile devices [144, 176]. Several solutions have been proposed.

Shevade et al. propose a pairwise tit-for-tat mechanism where node A forwards data for node B in proportion to the amount of data that node B has previously forwarded for node A [166]. A signed delivery ack serves as proof that the work was done by a next hop. Each node maintains the waiting time on links between itself and other nodes. Each node also computes the link capacity with other nodes and periodically floods these metrics throughout the network. This work assumes that node link-state is disseminated faithfully. Selfish nodes could easily attack the protocol by disseminating a modified link-state vector that is favourable to themselves. Increasing the inter-contact time or reducing the link capacity with other nodes would make the selfish node a less favourable node in the path. Selfish nodes may also drop delivery acks package in an effort to artificially skew their own participation in a successfully delivered message. This routing scheme also assumes that messages have uniform value and consequently does not favour any message over another. Source routing is also incompatible with a CTN. Source routing would not only require that content holders care about the demands of others, but for intermediate nodes to comply with the source node chosen route. Moreover, the source node is not guaranteed to know the identities or contact behaviour of all intermediate nodes or even the destination node. This protocol is therefore not suitable for use in a CTN.

Social Selfishness Aware Routing (SSAR) considers a DTN node topology where all nodes share some measurable social relationship [106]. In SSAR, the higher the social relationship, the more willing a node is to forward messages for the other. SSAR uses a variant of the multiple knapsack problem for buffer management and to make forwarding decisions. Messages are prioritized according to their delivery priority, which decays as the distance from the source node increases. Nodes also prefer to save messages of equal distance from the source for recipients with a stronger social relationship. Unfortunately, SSAR does not consider free-riding nodes in its design. It assumes existing social relationships are sufficient to ensure that nodes behave according to the specified routing algorithm. This assumption violates the goals of a CTN. SSAR also assumes the social relationship between two nodes is expressed honestly. While it has been shown that people can quantitatively

rate their friendships [58], we do not believe that this would work in practice due to the strict requirements for both geographical collocation and pre-defined social relationships.

Yin et al. propose a simple game-theoretic approach to routing in a DTN called Pay-for-Gain (PFG). Their approach is pure pairwise tit-for-tat scheme where source nodes buy buffer capacity in the network. PFG assumes that paying for buffer space guarantees that a message will be forwarded. In practice, selfish nodes could simply accept the credit and never deliver the message - i.e. mimic the behaviour of simply never encountering upstream nodes. Finally, MobiCent by Chen et al. introduce a trusted third party to provide centralized accounting between DTN nodes [30]. Like COMBINE, OCMP, and BitTorrent, MobiCent's use of a centralized component invalidates it for use in a CTN.

The work surveyed in this section provides significant insight into the design of DTNs. The work examines the expected behaviour of several DTN routing protocols and their expected performance. However, the assumptions in this body of survey work differ from those underlying CTNs. Most of the work surveyed in this section is designed for *managed* DTN systems, where nodes in a deployment are managed by a single authority and are assumed to well behaved.Moreover, managing authorities are typically more concerned with the aggregate performance of the system rather than the QoS for any single node. We strongly believe that the software implementing CTNs must be open source, and freely available to everyone. Under this design assumption, none of the work described in this section is suitable for use in a CTN. By releasing the code, all protocols can be manipulated by would-be free-riders. For CTNs to be practical, we require a DTN routing protocol that is resistant to free-riders.

2.3 Chapter summary

This chapter has surveyed a broad range of work generally related to the creation of a CTN. Despite an enormous amount of related work, we have shown that the necessary building blocks of a CTN do not exist.

Chapter 3

Design Principles for Opportunistic Communication in Constrained Computing Environments

In the previous chapter we surveyed a broad range of related work and demonstrated that existing research is insufficient to achieve our goals and piece together a censorship-resistant communication network. Before describing the architecture and design of a CTN, this chapter briefly summarizes a set of design principles that influence the design of our prototype CTN implementation.

3.1 Introduction

CTNs are designed to operate using personal mobile devices. These computing platforms are characterised both by their constrained mode of operation and distinctive physical attributes. As devices used extensively for communication and other personal tasks, maximizing battery life is a key concern. To conserve energy, these devices use low-power CPU(s), which are often underclocked to reduce heat and maximize battery life. They typically contain a limited amount of RAM; roughly an order of magnitude less than an average personal computer. Persistent storage is provided by low-power solid-state Flash memory. Short range wireless technologies such as Wi-Fi and Bluetooth are also standard. As the dominant form of future computing [92] and the target computing environment for CTNs, understanding and adapting to the trade offs they present between computing and communication resources and energy consumption will become increasingly important.

Although today's cellular networks provide nearly ubiquitous data service, using these networks to communicate with centralized servers makes it easy to track and censor mobile users [150]. Instead, devices operating in a CTN must utilize intermittent, short-lived, wireless *opportunistic* connections to exchange data with other mobile devices. The capacity, or throughput, of these connections is further handicapped by their *constrained computing environments*. In this chapter we consider the effect of a constrained computing environment on opportunistic communication. Moreover, we argue that a conventional system model, where applications operate oblivious to their connectivity state, is inefficient in an opportunistic communication setting. Based on our experiences with two related systems, we propose a set of design principles for CTNs and other systems that use opportunistic communication.

We derive our design principles from our experience with two real-world systems introduced in Chapter 2. The first system, KioskNet, uses opportunistic connections between a

kiosk-based computer and an embedded device in a vehicle to transport data to and from kiosks in rural areas of developing regions to gateways on the Internet [66, 67].The second system, MobiClique, exploits human mobility to provide decentralized store-and-forward communication between mobile devices [143, 144]. Although these systems solve different problems and cater to very different types of users, we believe that they encapsulate the problems that other similar systems would face in that their effectiveness is directly dependent on their ability to maximize communication during opportunistic connections.

This chapter begins with a brief overview of the KioskNet and MobiClique in Section 3.2. In Section 3.3 we define our system model and outline the set of system constraints that adversely affect opportunistic communication. Based on our experiences, we define a set of design principles for opportunistic communication in Section 3.4. We conclude in Section 3.5.

3.2 Overview of existing systems

The following two systems utilize opportunistic wireless communications to exchange data amongst resource-constrained, wireless-enabled devices.

3.2.1 KioskNet

KioskNet is a mobile system that provides very low-cost Internet to developing regions. Building on the pioneering lead of Daknet [137], KioskNet utilizes buses and cars as "mechanical backhaul" devices to carry data between rural village kiosks and Internet *gateways*. A village kiosk consists of a kiosk *controller*, a low-cost, low-power embedded computer that can be powered from an independent power source such as a solar panel. Villagers access the kiosk using a recycled PC connected to the controller. Data created by users, such as emails, information requests, videos, etc., is fragmented and stored on the controller as self identifying *bundles*. A controller is assumed to have Wi-Fi, and possibly a cellular or dial-up connection. Although controllers can communicate with the Internet using a variety of connectivity options, mechanical backhaul is the primary mode of communication. Mechanical backhaul is provided by cars, buses, motorcycles, and trains that pass by a kiosk and also pass by an Internet gateway. Such entities are known as *ferries*.

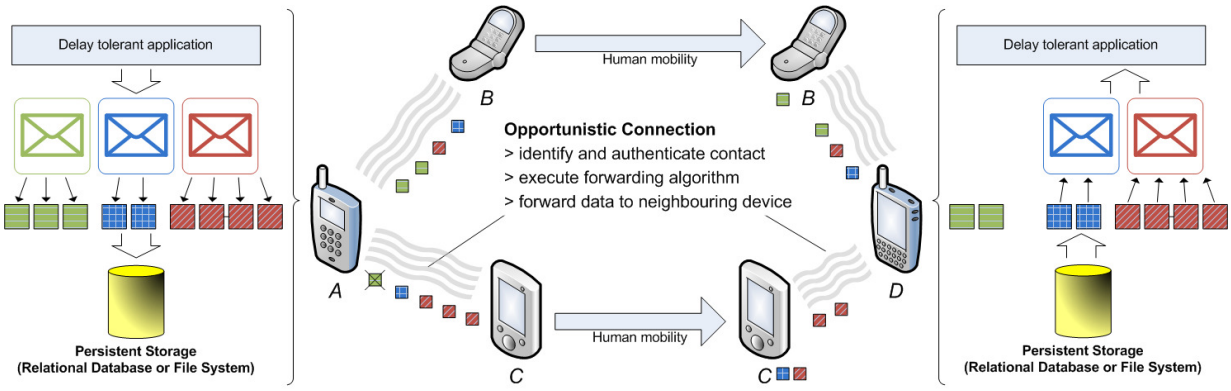


Figure 3.1: Example scenario: Mobile devices operating in a pocket-switched network.

During an opportunistic connection with a controller, which may last from 20 seconds to 5 minutes, bundles are transferred from the controller to the ferry. Similarly, bundles destined for a kiosk user is transferred from the ferry to controller. To minimize redundant bundle transfers during an opportunistic connection, each connection begins with a meta-data exchange where each component exchanges a list of its existing bundles. Only bundles that have not present on the other component are transferred. Each component maintains a record of how many times each bundle was forwarded, which ensures that precedence is given to bundles that have been forwarded the fewest times.

Similarly, ferries upload and download bundles opportunistically to and from an Internet gateway, which is a computer that has a Wi-Fi interface, storage, and an always-on connection to the Internet. The gateways are likely to be present in cities having DSL or cable broadband Internet access. A gateway collects bundles opportunistically from ferries and stages them in local storage before uploading to a server on the Internet. It also downloads bundles from the Internet on behalf of kiosk users, and transfers them opportunistically to the appropriate ferry, governed by a routing protocol [68].

3.2.2 MobiClique

As a form of pocket-switched network [76], MobiClique exploits natural human mobility and opportunistic wireless connections to disseminate data from device to device across an otherwise disconnected network. Unlike KioskNet, where data flows to and from rural

kiosks and Internet gateway, each device running MobiClique is both a source and destination for data content. The system depends on intermediate devices to ferry data between source and destination. An example scenario is illustrated in Figure 3.1. Delay tolerant applications such as email pass data into MobiClique as opaque objects. The data is subsequently fragmented into smaller bundles and stored in persistent storage (a relational database). At the receiving device, bundles are stored, reassembled, and passed to the receiver's instance of the application.

Devices running MobiClique operate by continuously scanning for neighbouring devices through a combination of Bluetooth scans and UDP beacons sent over Wi-Fi. When another MobiClique device is detected, an opportunistic connection is established. During an opportunistic connection, each device exchanges a set of identifying/authenticating information and metadata about the user. In MobiClique, this metadata consists of the user's social profile: their friends and their interests. Each device uses this metadata to query a local relational database that returns a set of bundles that should be forwarded to the neighbouring device. Referring to our example figure, device A would have determined that device B and C were friends or shared interests with device D, and were thus likely to deliver data to D.

Like KioskNet, MobiClique's ability to efficiently disseminate content is dependent on its ability to maximize data transfer during an opportunistic connection.

3.3 Computing environment

With an overview of two existing systems as context, we now present a more detailed system description and consider the constraints that inhibit opportunistic communication.

3.3.1 System model

Opportunistic communication is typically provided at the OSI session layer. Applications access this layer through a conventional set of APIs for sending and receiving data. Data received from applications is fragmented into bundles. These bundles are then stored in persistent storage to provide robustness to power failure during periods of long disconnection. Bundles received by other devices are also stored in persistent storage. When all

of the bundles composing a data item are received, the data is passed to the application layer.

Associated with each bundle is a set of metadata. Metadata is application dependent; however, it typically consists of the source and destination of the bundle, a creation timestamp that is used to expire bundles, a globally unique identifier, and value identifying its structure or purpose.

We make three fundamental assumptions regarding the data and metadata handled by this layer.

- **Metadata fits in memory**: Although pathological metadata schemes can be defined that consume vast amounts of memory, we assume that metadata can always fit into main memory. This assumption is easy to satisfy even in low-memory environments; because by doubling the size of a data bundle, we can usually reduce the total metadata size by half. Metadata per bundle is about 100 bytes at most.
- **Application data bundles cannot fit in memory**: For these systems to scale to a large number of users, they must accommodate large amounts of data for a wide range of users. For example, two hours of music recorded at 192 kbps (which is currently typical) occupies about 170 MB. Twenty-minute videos can range from 20 MB to several hundred MBs. Given that mobile devices have about 1 GB of RAM, we assume that the amount of data contained on any device for the purpose of forwarding exceeds the amount of main memory and must reside on persistent storage.
- **Every opportunistic connection needs different data**: To disseminate large amounts of data, devices should not repeatedly transfer the same bundles during each connection. For example in KioskNet, connections between ferry and rural kiosk must exchange bundles that have been forwarded the fewest number of times. In MobiClique, the bundles exchanged during a connection depends on the identity and social relationship with the neighbouring device. Each opportunistic connection must therefore identify the neighbouring device and making a subsequent bundle selection decision.

The primary function of the opportunistic communication layer is to establish and participate in opportunistic connections with other devices. These connections are generally

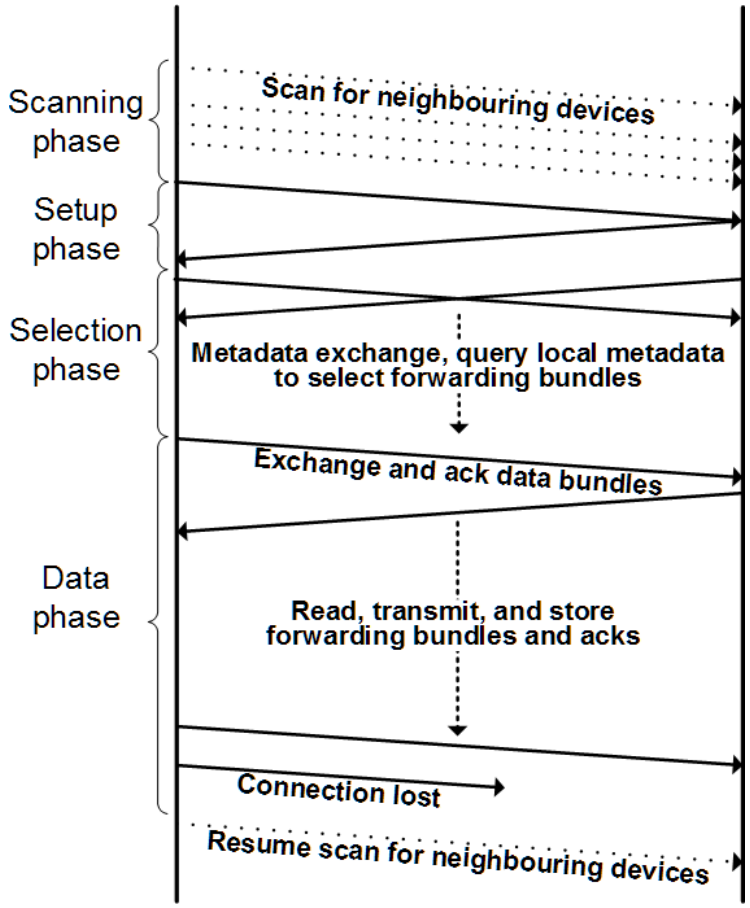


Figure 3.2: Phases of an opportunistic connection.

defined by the following four phases:

- **Scanning phase**: Each device continuously scans for neighbouring devices. The scanning interval may be static or dynamic depending on the user's previous mobility and current context. Scanning may be performed passively by listening for beacons or actively by probing for neighbours.
- **Setup phase**: Upon detecting and connecting to an opportunistic contact, each device must first identify the contact, and in some cases, authenticate and setup a secure connection.

- **Selection phase**: After establishing a connection with the neighbouring contact, each device must select a subset of its local bundles to forward. In sparse networks, where opportunistic connections are infrequent, this selection may be simple. For example, in KioskNet, bundles travel either upstream to the Internet, or downstream to a rural kiosk. The selection phase therefore only considers the class of contact (i.e. kiosk or gateway) and selects bundles in the order of least times sent. In dense networks, where devices are in frequent contact, more complex queries are necessary. The selection phase in MobiClique selects forwarding bundles based on their destination and social relationship with the contact. To prevent redundant bundle transfers, MobiClique maintains a record of each bundle successfully transferred to each device in persistent storage. Experimentation has shown that the resulting selection phase can quickly grow to the order of tens of seconds because of the need to access persistent storage [143].

- **Data phase**: After selecting a set of bundles to be forwarded, each device begins the forwarding process. Ideally, this phase should occupy the largest portion of the opportunistic connection. During this phase, bundles are read from persistent storage, transmitted to the neighbouring device, and stored persistently on the receiver. At this time, the receiver either explicitly or implicitly acknowledges (depending on the transport protocol) the receipt of each data bundle. The data phase terminates when both parties have finished transmitting data or when the connection is disconnected (either intentionally or due to signal loss).

The four phases of an opportunistic connection are illustrated in Figure 3.2.

Scanning and setup phases have been extensively studied in the literature [70, 192]. Scanning more aggressively reduces the expected delay in detecting a nearby device, at a cost of increased energy consumption. The trade off between scanning periods and missed opportunities is detailed in [192]. Similarly, the setup phase offers little opportunity for improvement and has been studied in great detail [70]. In existing systems, contacts are identified by their MAC address or an exchange of globally unique identifiers. Techniques for authentication and setup of a secure channel range in complexity; however, these are typically CPU-bound operations with a constant running time.

The remainder of this chapter therefore focuses on minimizing the duration of the selection phase, and maximizing the quantity of data transferred during the data phase.

Unfortunately, these phases are inhibited by a series of system constraints.

3.3.2 System constraints

Opportunistic communication is inhibited by the following constraints.

- **Need to limit energy consumption**: The need to limit energy consumption is a constraint that fundamentally defines constrained computing environments. On mobile platforms, conserving battery life requires low-power and sometimes underclocked CPUs, wireless technologies that cannot operate at full data rates and frequently enter power saving modes [15], and low-power, high-latency forms of persistent storage. These hardware requirements also exist in embedded environments that depend on passive cooling and must minimize heat.

  This constraint affects decisions made when implementing an opportunistic communication system. As mentioned above, the chosen scanning frequency has a direct effect on battery life. Excessive computation causes the CPU to be active when it can otherwise be off. Finally, the choice of wireless interface(s) can significantly affect power consumption [138, 142].

- **Underpowered CPU**: Mobile devices are commonly designed with an underpowered and underclocked CPU to save energy and reduce heat. Other devices, such as Apple's iPhone [3], contain ample CPU power; however, the CPUs are disabled or placed in a low-power mode when the user is not interacting with the device. Underpowered CPUs affect three phases of an opportunistic connection. (1) For systems that must authenticate contacts or establish a secure connection, an underpowered CPU increases the duration of the setup phase. (2) In the selection phase, each device must execute a forwarding algorithm to select a subset of bundles to forward. In KioskNet and MobiClique, this selection is computationally simple; however, other forwarding schemes have been proposed that demand significant computation [193]. (3) Finally, in both the selection and data phases, underpowered CPUs have been shown to be the primary bottleneck when performing wireless network I/O, and significantly reduces throughput during an opportunistic connection [128]. Measurements in KioskNet show that an underpowered CPU can reduce the maximum throughput of a Wi-Fi card capable of 54 Mbps to 12 Mbps or less.

Constraint	Design principle
Need to limit energy consumption	Minimize redundant wireless data transfer
Underpowered CPU	Explicitly distinguish between periods of connection and disconnection
Poor and intermittent communication	Use hysteresis
Slow persistent storage	Cache metadata
Limited RAM	Maximize the use of available memory and adapt to low-memory conditions

Table 3.1: Design principle satisfying each system constraint.

- **Poor and intermittent communication**: The mobility of devices operating in an opportunistic setting introduces inherent communication problems. Devices may be detected on the fringes of communication range, which causes subsequent communication to be unreliable and problematic; particularly for control packets that initiate the connection [70, 208].

  Moreover, mobile devices that are in communication range may eventually move out of range. Depending on the wireless technology used, communication interrupts may cause random backoffs, inordinately long delays, and failures that can hinder other connection opportunities. Devices need to somehow determine that the communication opportunity has ended, and cleanly close open descriptors. Moreover, they need to deal with cases where the opportunistic connection resumes after a brief pause.

- **Slow persistent storage**: Low-power persistent storage takes two forms: disk drives that operate with a low disk RPM and spin down when not used, and Flash memory. Slow persistent storage affects both the selection and data phases. Slow (random) I/O causes the retrieval of metadata to take order of magnitude longer than if the metadata was in memory. Slow I/O also serves to reduce the overall throughput of a connection. We have found that reading and writing to slow persistent storage (hard disk and Flash memory) during an opportunistic connection reduces wireless throughput by approximately a factor of two [128].

- **Limited RAM**: Despite the continuous decrease in memory costs, RAM continues to be a highly constrained resource on mobile devices. Constraints due to limited

RAM manifest themselves in two forms and inhibit both the selection and data phases. First, lack of RAM prevents applications from storing application data in memory and thus requires accessing slower persistent storage during an opportunistic connection. Second, querying, reading, and transmitting data during an opportunistic connection typically results in a rapid series of memory allocation requests that often consume all of the available memory and cause virtual memory swaps that further slow access to persistent storage. Subsequent allocation requests during the opportunistic connection must block waiting for the virtual memory swap to finish.

The problems stemming from limited memory are magnified by virtual machines such as Java and .NET, which are becoming increasingly common on mobile devices. Both Java and .NET rely on garbage collection to reclaim memory. Garbage collection is known to be computationally expensive [209]. Although garbage collection can be performed eagerly during periods of inactivity, it is typically performed lazily and reclaimed on demand. Performing garbage collection during an opportunistic connection can significantly reduce throughput.

3.4 Design principles

With an understanding of the parameters that inhibit opportunistic communication, we now present a set of design principles. The relationship between system constraints and design principles is summarized in Table 3.1.

- **Cache metadata**

  Storing metadata in main memory rather than persistent storage provides a quick remedy to the effect of slow I/O to persistent memory. Measurements in KioskNet show that even an in-memory naïve metadata implementation can reduce the duration of the selection phase by approximately a factor of five. Unfortunately, while we assume that metadata can always fit in memory, we cannot guarantee that memory will not be consumed by other applications or persist after a device reset. Applications should therefore treat in-memory metadata as a cache that provides hints during the selection phase [181]. An invalid hint may have the side effect of forwarding redundant data; however, successful hints will significantly increase performance.

- **Use all available free memory, but adapt to low-memory conditions**

  This principle follows naturally from the need to cache and minimize interaction with slow persistent storage during an opportunistic connection. The ability to determine the level of available free memory exists in nearly all smartphone and embedded OSes. Applications should proactively populate their in-memory cache when memory is unused.

  While opportunistic communication systems should consume as much main memory as available, they must be equally ready to release it back to the underlying OS immediately after use. In virtual memory environments, failure to release memory back to the OS will eventually result in frequent page swaps. In environments without virtual memory, memory allocations will either fail or emergency garbage collection will take place[1]. We have found that low-memory problems frequently occur during opportunistic connections when large amounts of memory are requested.

[1]The default behaviour in most Java VMs is to wait until free memory drops below a predefined threshold before triggering garbage collection.

On recent smartphone platforms, including BlackBerry [6] and Windows Mobile [5], applications can register for low-memory events from the underlying OS and proactively release memory. While every non-trivial application should utilize these OS functions, their use is critical in an opportunistic communication setting.

- **Minimize redundant wireless data transfer**

  The most effective means of minimizing redundant data transfer is to avoid retransmitting bundles. Although this principle may seem obvious, its design and implementation is non-trivial and must be considered in the early stages of software design. In KioskNet, redundant data transfer is reduced through the use of a metadata exchange. Each connected component exchanges a list of bundle IDs identifying the bundles that it contains. Each component then requests only bundles that it does not have. In our current implementation, this exchange causes the selection phase to last approximately 30 seconds for a 100 MB workload and nearly zero bundles are redundantly transferred. MobiClique also minimizes redundant transfers without a metadata exchange; the selection phase consists purely of a complex database query and no network I/O. Each MobiClique device maintains a record of which bundles have been successfully delivered to each device; only unsent bundles are forwarded. This operation requires a four-way database join and can take over 30 seconds to select less than 5 MB worth of bundles. Clearly, the design of this optimization can have a significant impact on the performance of the system.

- **Explicitly distinguish between periods of connection and disconnection**

  We strongly believe that exploiting periods of disconnection can yield substantial performance gains. Although neither of our systems exploit this property to its full potential, we foresee using disconnection periods to (1) refresh the metadata cache, (2) compress/decompress data bundles, (3) pre-compute forwarding strategies, and (4) performing explicit garbage collection.

- **Use hysteresis**

  Poor and intermittent communication can cause connections to come and go frequently. Reacting to lost and re-established connections requires each device to enter the setup and selection phases before it can resume sending data. A system that uses

opportunistic communication should not immediately react to the loss of a connection; the connection may resume after a short delay. When a connection is thought to be lost, we have found that waiting 10 to 20 seconds before declaring it closed works well.

3.5 Chapter summary

Drawing on our experiences with two existing systems, KioskNet and MobiClique, we have discussed how the dominant form of future computing both enables, while its defining characteristics inhibit, opportunistic communication. By examining each constraint in a mobile device and its effect on opportunistic communication, we have enumerated a set of design principles. Although these principles may seem like common sense, they are not obvious when initially designing a system. We have learned them only by examining the flaws in our own work and have re-implemented parts of both systems to correct these problems.

In summary, we believe that developing applications for a resource-constrained environment requires careful consideration of how and when memory is used and storage is accessed, when a connection should be made and eventually closed, and most importantly, how energy consumption can be minimized; developing in these environments requires a sense of minimalism. We apply these principles to the design of a prototype CTN application in the next chapter.

Chapter 4

Censorship Tolerant Networking

In this chapter we present the high-level architecture of a CTN. Guided by the design principles outlined in the previous chapter and several software goals, we describe a software architecture for a prototype CTN system in Section 4.2. We present the details of our implementation and integrated experiment framework in Section 4.3.

4.1 Architecture

We approach the design of a CTN from the bottom up, starting with the need to provide private communication between participants. Existing DTNs achieve communication privacy through a hierarchical PKI, rooted at a trusted system administrator [163]. This approach clearly violates several CTN requirements. Hierarchical PKI schemes require a centralized authority to issue certificates and bootstrap trust. All participants in the system must know the certificate authority: both legitimate and malicious. This requirement creates a central point of legal and physical attack. Although multiple trust hierarchies could be created in an attempt to circumvent such attacks, disjoint groups of users belonging to different trust hierarchies would not be able to communicate reliably. Finally, a central authority, if compromised, would quickly erode the censorship-resistant properties of the network. Robustness would be lost by definition. Privacy could be lost if fraudulent certificates are issued. The system would cease to scale, as new members could not join the network. Moreover, any attempt to join the network would quickly reveal the identity of the prospective participant.

To satisfy the CTN criteria, the architecture must be fully decentralized. Nodes in our system operate by forming trust relationships with other mutually trusted nodes. Trust is expressed by directly disclosing an identity to another node. Our architecture leverages the trust relationships to create a simple, efficient scheme for organizing and expressing demand for content contained within the CTN. These concepts are detailed in remainder of this section.

4.1.1 Identity

A self-generated globally unique ID (GUID) and a corresponding self-generated public/private key pair uniquely identify nodes operating in a CTN. The private key is always

kept private by the node, and used to respond to authentication requests and for symmetric key exchange much like in TLS/SSL [43]. The GUID and public key are distributed to trusted nodes along with several other attributes that may change over time. These attributes include one or more wireless MAC addresses used for device discovery, an optional phone number for the device, and an optional human-readable alias. The role of the phone number will be explained later in this chapter and is necessary for the work presented in Chapter 6.

A CTN node does not share its identity with nodes that it does not trust. Sharing MAC addresses with untrusted nodes would allow them to de-anonymize CTN participants, and thus compromise their identity privacy. The only attribute that a node shares with non-trusted devices is the GUID. We assume that trusted nodes would not readily share identities other nodes due to the real-world social relationships that underpin the formation of the initial trust relationship.

4.1.2 Trust relationships

Sharing censored, potentially illicit content with another person requires trust. We therefore assume that a CTN node will only ever communicate with nodes whose users are mutually trusted. Our approach operates by forming pairwise trust relationships between nodes whose users trust each other in the real world. Relationships are formed by invitation through direct social interaction. By forming a trust relationship with a node whose users trust each other in the real world, we assume that the node will never betray the trust. That is, we assume that a node will never willingly disclose the identity of another trusted node. This assumption is fundamental to the design of a CTN. We will discuss exceptions to this assumption in Section 8.1.3.

Constraining communication to specific trusted nodes is a technique borrowed from our prior work with the MobiClique system [144]; however, recall that this approach relies on existing online social networks to infer and bootstrap trust and is therefore not suitable for use in a CTN. In contrast, our simple approach to forming pairwise trust relationships facilitates incremental deployment. In the aggregate, each pairwise trust relationship forms an edge in the graph that we refer to as the *trust topology*.

4.1.3 Content demand and organization

Content in a CTN consists of photos, movies, audio, and other files present on a mobile device. Content may be retrieved from any source or produced locally, and is assumed to be highly diverse in both its character and physical size. The primary role of a CTN is to disseminate content from content holders to nodes that desire, or *demand*, the content. We further assume that the content is censored or illicit in nature since non-censored content can be acquired directly from Internet-based sources with significantly less effort and resource cost. This assumption is important because it requires that identities of participants and information about the content that they demand or possess be carefully managed. Recall from Chapter 1 that the identity of a user is the primary asset of interest to attackers. Therefore, non-trusted nodes can never be allowed to learn the identity of a node carrying potentially illicit content.

We make two additional practical assumptions regarding user demand for content. First, we assume that users' preference for content is highly diverse. This property has been observed in prior work studying personal music collections [98], online video consumption [59], and BitTorrent downloads [203]. Second, we assume that demand can greatly exceed the storage capacities of nodes and communication capacity of the network. The underlying routing protocol must therefore be designed to discriminate amongst many concurrent requests for the node's resources, while maintaining compliance with the CTN design criteria outlined in Chapter 1.

Demand for content

Nodes in existing mobile content sharing systems express demand for content through one of two means. The first approach is to express demand for content through strategically chosen keywords [76]. By disseminating keywords of interest to other nodes in the network, nodes, in theory, retrieve content with matching keywords. This method has three major flaws. First, a keyword-based scheme implicitly the taxonomy of the content to be the entire lexicon of every language. Second, it requires that users tag every item, an onerous task. Third, even if the content is optimally routed from source to destination, many items may match a node's keywords of interest. An incorrect selection of content by intermediate nodes costs each node both energy and storage, and thus reduces the transmission capacity

of the network. In the absence of malicious tagging, this method may work for popular content. For example, the keywords "tiananmen square tank man" unequivocally corresponds to the historical Chinese image. However, the multitude of keywords corresponding to any content item easily illustrates the inherent inefficiencies of a keyword-based selection scheme.

Other mobile systems mimic the publish-subscribe model and organize content according to a pre-determined *channels* [16, 112]. Nodes subscribe to a channel and retrieve constituent episodes from neighbouring devices carrying the desired channel content. Given that the set of all content is extremely large with highly diverse producers, a static hierarchy would clearly not be scalable.

Both the keyword-based and publish-subscribe methods are a form of late binding between nodes and the content that they demand. A node that has demanded content will be unaware of the specific content item(s) retrieved in satisfaction of their request until the content is eventually delivered. In addition to the inefficiencies and poor scalability of these schemes, we believe that this unavoidable ambiguity is sufficient grounds to dismiss both approaches as impractical for use in a real-world system.

Our approach leverages the trust topology to provide a simple, efficient scheme for expressing demand for content explicitly. Our approach is unique in that it tightly couples nodes' demand for content with the specific items being demanded within a known, finite *Content Space*, defined next.

Content Space

The Content Space, CS, is the continuously growing set of all content in existence in the system. We will refer to the elements of CS generically as *content items*, which are uniquely identified by a Content ID (CID) derived from a hash of its contents. Several metadata fields accompany content items. These are: a size, a content type, a filename name, and human-readable description.

Constraining communication to an explicit set of trusted nodes creates a convenient bound on the information that must be maintained about other nodes in the system. We use this constraint to define CS_i as the view of all content known to exist by node n_i. During an opportunistic connection between two nodes, each node *may* exchange metadata

corresponding to content from one or more sources. The first source of metadata is from local content. Local content is content that exists on a node that is either partially or completely retrieved from other nodes. A metadata exchange *should* also include the metadata for content known to exist on other nodes that are *reachable* by the node. Reachability is defined in Section 5.2.4. These metadata updates also include the GUID of the node that owns the content.

The protocol underlying the exchange of content during an opportunistic connection will be explained in detail in Chapter 5. At this stage in our discussion, it is important to note that sharing metadata is not a strict requirement for participation in a CTN. For example, nodes electing to never transport content on behalf of others may simply restrict their metadata transmission to locally-held content. However, not sharing metadata prevents retrieving metadata for others, which we will show is a necessary condition for long-term participation within a CTN with nodes that are sensitive to resource costs.

Each node expresses demand for content by explicitly listing the CIDs of demanded items in its *demand vector*. Content items are assumed to be large enough to exceed the link capacity of opportunistic connections between nodes and therefore must be transported across the network as *fragments*. Each element of the demand vector contains a bitmap of fragments that have already been retrieved.

Each node maintains a mapping of node GUIDs that own (or partially own) content items to the corresponding content metadata. This mapping contains GUIDs for both trusted and non-trusted (anonymous) nodes. However, metadata corresponding to content that is partially owned is only maintained for trusted nodes. This mapping facilitates efficient lookups of both the content owned by a specific node. A reverse mapping between the CID and node GUIDs is also maintained to allow efficient searches of nodes that own specific content items.

In practice, a node may add or delete items from the device at any time. A node's Content Space may therefore become stale over time. To overcome this situation, we associate a timestamp with the metadata corresponding to every non-local content item. The timestamp is updated every time the metadata is detected. The timestamp is subsequently used to delete metadata based on age. For example, suppose that node A frequently contacts node B. When A first contacts B it retrieves all of B's Content Space. Each association of a content item with B includes the current timestamp. Node B subsequently deletes some content. Since node B does not maintain a record of content metadata known to

A, it is not expected to notify A of content deletions. Instead, during a subsequent metadata exchange, A updates the timestamp of content observed on B with the current time. When A's storage capacity becomes constrained and it needs to prune the Content Space, A should delete the content items with the oldest observed timestamp first.

The Content Space may grow exponentially as new nodes enter the system and trust relationships expand. Our current prototype implementation prunes the Content Space based on a least recently encountered policy: when the Content Space grows beyond a user-specified threshold, the metadata for content belonging to the least recently encountered node is deleted. Metadata for content on non-trusted nodes that, by definition, have never been contacted inherits the contact statistics of intermediate trusted nodes. Alternative strategies for pruning the Content Space can also be implemented in our system, and is discussed as future work in Section 8.2.1.

4.2 Software architecture

We motivate the software architecture of a prototype CTN system by first outlining several high-level goals that influence our software design and implementation. These goals refine the CTN design criteria outlined in Chapter 1 and the design principles outlined in the last chapter.

- **Platform agnostic**: Unfortunately, the software APIs exposed by today's mobile software platforms are highly heterogeneous and non-standardized [125]. The prototype CTN software must accommodate this heterogeneity to the greatest degree possible.
- **Hardware agnostic**: Today's mobile devices are commonly equipped with Wi-Fi, Bluetooth, and cellular radios, with different capabilities, usage paradigms and corresponding APIs. The logic underlying the CTN software should be agnostic to changes in underlying hardware.

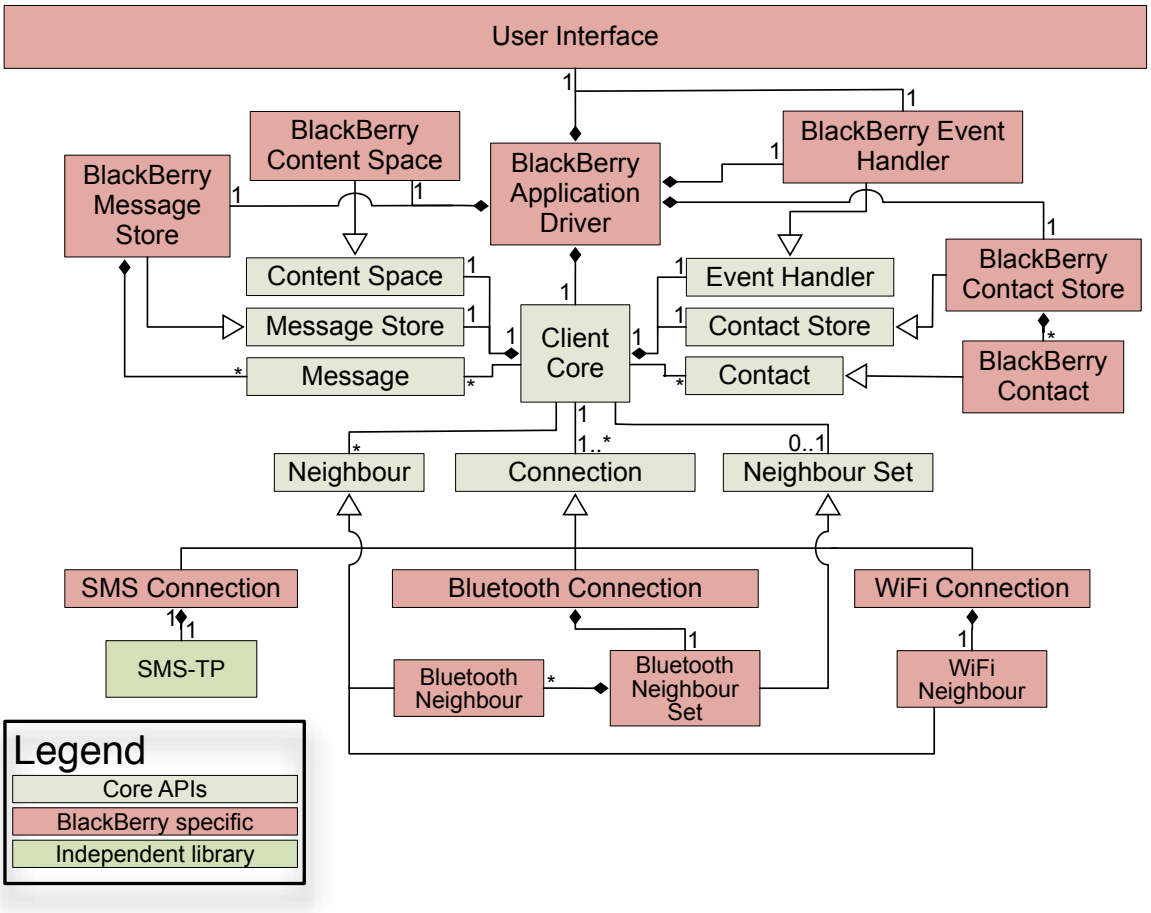


Figure 4.1: Software architecture of the BlackBerry CTN prototype.

4.2.1 Components

This section describes the major components of our prototype CTN system. The software architecture of the system is illustrated in Figure 4.1.

Connection

Connection objects reside at the lowest level of the CTN stack. A Connection object is an abstract representation of a network interface. The Connection API encapsulates two

sub-APIs. The Neighbour API is an abstract representation of a neighbouring device. The device may be any device and not necessarily a participating node in the CTN. The Neighbour object encapsulates the information needed to establish a connection with that device. This information may include a MAC address of a neighbouring Bluetooth device or an IP address of another Wi-Fi-enabled device. The Neighbour Set API is an abstract representation of a collection Neighbour objects.

Connections have two overlapping operating modes. The first, neighbour detection mode, is responsible for initiating neighbour discovery at a specified frequency. This may include transmitting UDP beacons, or doing a Bluetooth device inquiry scan. After detecting a set of neighbouring devices, the Connection object passes the corresponding Neighbour Set up the stack to the Client Core through a neighbour detection event. The Client Core is the heart of our prototype implementation, and will be described later in this section.

Connection objects that are capable of detecting neighbouring devices are able to initiate and accept connections to and from other devices (assuming that the neighbouring device is still within communication range). Outbound connections are initiated by the Client Core by calling `connect()` on a Neighbour object. Upon receiving a connection, the Connection object constructs a single Neighbour object that encapsulates the connection. This Neighbour object is passed up the stack to the Client Core as a neighbour connection event. Once a connection is established between a pair of CTN nodes, the Neighbour class allows communication with the connected neighbour through conventional `send()` and `recv()` primitives.

Connection objects that are capable of detecting neighbouring devices may create Neighbour objects without having to first detected them nearby. That is, given a set of data needed to connect to a neighbouring device, a Neighbour object can be created to encapsulate the data. For example, if the WLAN IP address of a neighbouring node is known, then a Neighbour object can be created by the Wi-Fi Connection object. This function of the Connection API is essential to handing off communication with a neighbouring node from one interface to another. The most common hand-off being between Bluetooth to Wi-Fi. This process is discussed in Section 4.3.2.

Connection objects may optionally register themselves as operating in *control channel mode*. The architecture permits any interface may operate as a control channel. However, in practice, messages sent on control channel interfaces should have much lower end-to-end

delay than messages transmitted through an opportunistic delay tolerant network. Like the process of accepting a connection, any control messages received by a control channel are passed to the Client Core for subsequent processing.

Client Core Message

Control messages exchanged between pairs of nodes are essential to managing the state of the trust topology that underlies the CTN. The Client Core Message API provides a common interface to producing and consuming these control message.

A Client Core Message has several attributes: the GUID of each of the source and destination nodes, a message type, a length, a variable length payload, and a delivery deadline. The presence of a delivery deadline stems from the fact that control messages are delay tolerant and may be delivered to recipients over any available network interface. Depending on the available interfaces, messages may be delivered opportunistically through short-range wireless links or transferred over WAN links. It is the responsibility of the Client Core to schedule a control message for transmission over specific network interfaces to ensure that the delivery deadline is met.

Although control messages contain both a source and destination GUID that allow them to be delivered across multiple hops, we believe that in practice most control messages will be delivered via a single hop. We discuss how control messages are delivered in the next chapter.

Message Store

The Message Store is responsible for providing persistent storage of control messages. The Message Store, similar to the other data stores, exposes a *view* of its underlying data structures. Elements of the view are sorted by earliest delivery deadline, allowing the Client Core to efficiently determine if a delivery deadline has been passed.

Contact

A Contact is an abstract representation of a node in the system. Contact objects must maintain and provide access to elements of a node's identity. These elements include

any known MAC addresses, the node's GUID, and its public key. A Contact includes the node's human-readable alias and phone number. Contact objects also encapsulate all known data about the state of the node in the system, which is used for routing. This state includes the last time that the node was encountered, the estimated inter-contact period, (a common metric for ranking nodes [29, 90, 173, 204]), and the last time that various routing protocol functions were performed with the node. Finally, contact objects are responsible for serializing their contents to a standardized format for exchange between nodes running different mobile software platforms.

Contact Store

The Contact Store is responsible for maintaining a record of the nodes underlying secure, trusted communication in the CTN. These are encapsulated in the Contact objects. Like the Message Store, the Contact Store provides multiple views of the underlying trust topology. Nodes may be trivially sorted by the least recently seen node or by their alias. However, it is often useful to arrange nodes by their potential *value* to the user. We discuss strategies used by the Contact Store to rank nodes in the next chapter. Implementation considerations of views are discussed in Section 4.3.3.

Event Handler

The Event Handler provides a bridge between the Client Core and the user. The Event Handler handles all events that are not handled by the Client Core, which are mostly events that require user attention. The Event Handler also receives events generated by the user that need to be processed by the Client Core.

Like the data stores, the Event Handler also exposes a view of its internal state. The view is primarily used by the UI layer to view lists of pending events.

Content Space

The Content Space object is responsible for organizing and managing content in the CTN. The Content Space at a node has three primary functions. The first function is to maintain the demand vectors of all contacts know to that node. Conceptually, content exchange

takes place over the censorship-resistant communication substrate provided by the trust topology. Architecturally, the Content Space is tightly coupled with the Contact Store. Every node in the Contact Store has a corresponding demand vector in the Content Space.

The second function of the Content Space is to provide a mapping between node GUIDs and collections of content. These collections correspond to the content known to have existed on that node at some point in the past (it may have been deleted to free up space), and may represent both partial and complete content items. The Content Space also maintains the reverse mapping of content items to a set of owners. This mapping allows both users and the Client Core to efficiently determine the set of nodes that own a desired content item.

The third function is to monitor the local file system of the device for content. Content may be created, modified, or deleted at any time. Newly created content in the file system is securely hashed to produce its CID. The CID and metadata corresponding to the newly content are automatically added to the local collection of known content. The Content Space also maintains a mapping of filenames of local content to their corresponding metadata. In the event that a file is deleted from the device, the Content Store deletes the metadata corresponding to the removed file. However, if other nodes own the content item, then the metadata is preserved and only the local node's GUID is removed from its list of owners.

The Content Space exposes the most functionally diverse view of all the components. The Content Space view may filter content that is either local or remote. View can also sort content based on size, number of outstanding fragments, quantity of outstanding data, or the number of owners.

Client Core

The Client Core is the only major architectural component to have a concrete implementation. All other components discussed in this section are APIs only. Concrete implementations of each API are platform specific.

The Client Core has three key roles: to process Events or hand them off to the Event Handler, to implement the routing protocol between a pair of connected devices, and to ensure that control message delivery deadlines are met.

Events from the user are delivered to the Client Core via an event queue within the Event Handler. These events correspond to the operations by the user that result in the creation and subsequent transmission of a control message. These events may include text messages exchanged between users (illustrated in Figure A.4(d)) or state updates, which we describe as future work in Chapter 8. Events sent by the user are always assumed to be handled asynchronously by the Client Core and no response is returned to the user. Client Core Messages derived from User Events are scheduled for transmission by adding them to the Message Store with a specified delivery delay.

The second source of events comes from the Connection objects. As previously discussed, these events include neighbour detection events, neighbour connection events, and control message events. Neighbour detection events are accompanied by a Neighbour Set object that contains a Neighbour object for each neighbouring device. This event is handled synchronously by the Client Core and executes on the Connection object's single thread; thus blocking the neighbour detection process while the single thread is busy within the Client Core. Upon receiving a Neighbour Set, the Client Core selects a subset of Neighbours by querying the Contact Store with the physical address of the Neighbour. We refer to the subset of trusted Neighbour objects as a *candidate set*. The Client Core iterates through the candidate set and attempts to establish a connection with each Neighbour. Once a connection is successfully established with a neighbouring, trusted CTN node, both nodes initiate a pairwise, tit-for-tat protocol. Both the strategy for ranking nodes and the pairwise protocol are discussed in the next chapter.

Neighbour connection events signal that a connection has been made from a neighbouring node. The Client Core using the Connection object's thread also handles this event synchronously. We will justify this design decision in Section 4.3.2. A Neighbour object that encapsulates the interface-specific connection accompanies this event. The Client Core immediately drops connections from nodes that either have an unknown physical address, and are untrusted by definition, or fail to initiate a secure connection using the receiver's public key. If the identify of the connected node is known (and by definition trusted), and a connection is successfully established, then the pairwise communication protocol is established. As a convention, the node that initiates the connection also initiates the pairwise protocol.

Control message events received by a Connection object mirror those received from the UI layer. They currently only include text messages received from another node, which are

subsequently passed to the Event Handler for eventual consumption by the user.

The Client Core has a single utility thread that is responsible for performing tasks during periods of non-connectivity. This task periodically connects to each data store to perform cleanup tasks, which are implementation-specific. However, when a Neighbour Set is present and the Client Core is actively engaged in opportunistic connections with other CTN nodes, all functions of this thread are suspended except for a single check to the Message Store to determine the next control message deadline. If a deadline has expired, the utility thread retrieves the control message(s) and delegates them to the first Connection object that operates in control message mode.

User Interface

The user interface (UI) is not part of the CTN core architecture. Although the CTN can operate independently of a UI, the UI is essential for forming and managing trust relationships and expressing demand for content. As previously discussed, elements of the UI consume events from the Event Handler and pass new events down the stack via the Event Handler. UI elements are allowed to manipulate the underlying data stores through their respective APIs. UI elements visualize the state of each store by retrieving an abstract view from component. The implementation details of the UI for our prototype implementation is discussed in Section 4.3.3 and illustrated throughout Appendix A.

4.2.2 Communication protocols

Communication between two CTN nodes differs depending on the state of the devices' trust relationship.

Bootstrapping secure communication

Communication with another node requires trust. As we have previously discussed, trust is established through the mutual exchange of identities. These identities contain all of the information needed to both identify and initiate communication with a node. An identity can contain many attributes that uniquely identify a node. A telecom provider under order, for example, can easily and quickly map the presence of a phone number to an identity

from a repressive government regime. An identity must therefore be exchanged between two nodes over a private channel.

We bootstrap security in our prototype CTN through direct social interaction between two CTN users. When two, presumably mutually trusted, users contact each other and would like to form trust relationship, the invitee sets the device as operating in invitation mode. This mode temporarily allows incoming (encrypted) connections from unknown devices. The inviter then identifies the other in its current Neighbour Set. The inviter is then prompted through the Event Handler and UI to enter an integer value. This value is then verbally or visually shared with the invitee. The integer value is subsequently used by the node as a symmetric key to bootstrap an encrypted connection with the neighbouring node. Upon connection establishment, the invitee is prompted through the device UI to enter the integer value. If the value is correctly entered, the Client Core can decrypt the connection stream. After successfully establishing an encrypted connection, the nodes proceed to exchange identities. This signals the end of the bootstrapping phase.

This approach is a quick means to bootstrap a private communication channel to exchange identities. However, some elements of an identity can not be protected. A third party eavesdropper can easily discover hardware addresses by passively intercepting link layer frames. However, we believe that the use of encryption obfuscates the communication channel sufficiently to create plausible deniability that the communication session was not used for illicit purposes. The same argument applies to our standard communication protocol, which uses stronger encryption. We revisit the topic of security and privacy of our system in Chapter 8.

Standard communication

The mutual possession of identities allows two nodes to connect to each other without user intervention. This is the standard mode of communication in a CTN. All connections are initiated by the Client Core. For the purposes of discussion, we'll refer to the device that initiates the connection as the client and the other node as the server. The client begins by generating a random key, k_c. The key is then encrypted using the public key of the server, (the public key is part of its identity), and transmitted to the server over the channel. Assuming that the client's connection is accepted, the server using its private key subsequently decrypts the key. The server subsequently generates its own random key,

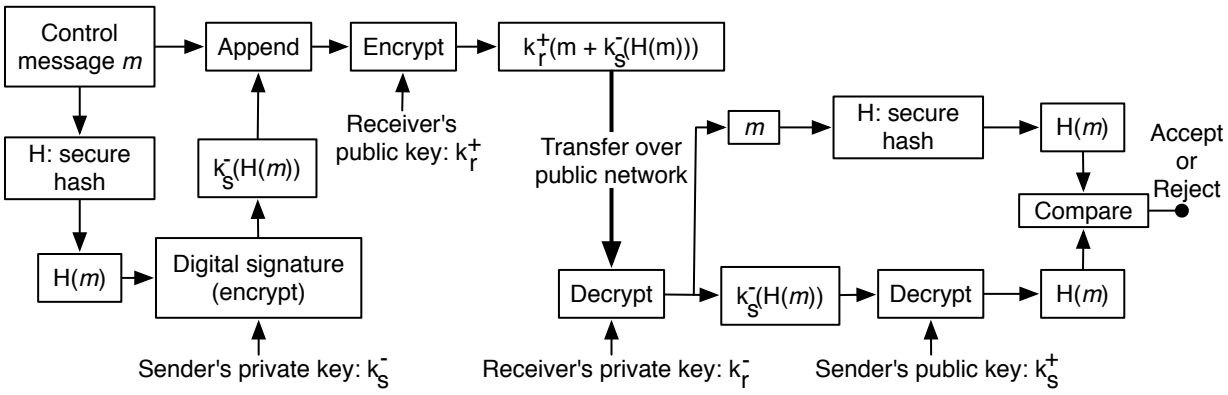


Figure 4.2: Control message encryption and decryption protocol.

k_s. This key is encrypted using the public key of the client and the server transmits the encrypted key back to the client. The client decrypts the transmission to reveal k_s. At this stage neither node has authenticated the other. Both nodes hold a version of k_c and k_s. If either party does not hold the required private key, then the decrypted key will differ from the source key. We therefore authenticate each node by producing a single shared symmetric key $k_{cs} = k_c \oplus k_s$, which is used to initialize an encrypted stream between the two nodes. Both nodes finish the setup process by transmitting a nonce to the other node encrypted using k_{cs}. If k_{cs} on both the client and server match, then the nonce will be successfully echoed back to the node. After the connection is successfully established, the two CTN nodes initiate the pairwise tit-for-tat protocol outlined in the next chapter.

Control messages are created by the Client Core and delegated to a Connection object for transmission. Control messages are intended to be transmitted asynchronously. The Client Core is only notified of a successful or failed delivery. However, privacy and source authentication of the control message is still very important. Prior to delegating a control message to a Connection object, the Client Core computes a digital signature of the message using its private key. The signature and the control message are then encrypted directly using the recipient's public key (control messages are limited to 32 KB in size). When the control message is delivered to the recipient node, the message is decrypted using the node's private key. Before accepting the control message, the node decrypts the signature using the source node's public key and verifies the secure hash of the plaintext control message. If the signature is rejected, then the control message is discarded and no response is returned

to the source node. The message is otherwise accepted and processed by the Client Core or the Event Handler. This protocol is illustrated in Figure 4.2.

4.3 Implementation

In compliance with our first software goal of platform independence, we chose to implement our software architecture in Java. Our core implementation is CLDC compliant [81] and therefore runs on a wide range of Java enabled devices and both Android and BlackBerry smartphones. Both the Android and BlackBerry platforms have the capability of running an autonomous CTN application [125]. We chose to implement device specific components of our prototype CTN on BlackBerry. Our implementation as of April 2012 is 32,245 non-comment lines of code and will be released under the Apache open source licence upon publication of this thesis. The Bouncy Castle cryptography library [182] used for encryption and authentication adds an additional 72,098 non-comment lines of code to the project. Unfortunately the poor performance of Bouncy Castle forced us to disable the use of encryption in both the experiments and field trial that we will discuss in the next chapter. Other system projects have reported similar performance problems with the open source library [87, 199, 207]. Given that approximately two-thirds of the code resides within the core, we estimate that an Android version could be easily written in approximately three weeks of development time.

The application runs continuously in the background of a BlackBerry device. The prototype has a user interface that allows the user to view the state of the Content Space, Contact Store, Message Store, and the contents of any current Neighbour Sets (i.e. neighbouring devices). Appendix A contains screen captures of the prototype CTN system.

4.3.1 Data storage

Persistent data storage in our prototype implementation takes two forms. The data underlying the Contact Store, Message Store, and Content Space is stored in memory resident data structures and flushed to Flash memory using the BlackBerry Persistent Object APIs [151]. Changes to the Content Store and Message Store are infrequent, so changes are immediately flushed to Flash memory. Changes to the Content Space data structure are

both frequent and take place primarily during opportunistic connections. Moreover, the data structure can become very large as the Content Space grows. Flushing the Content Space data to persistent memory during an opportunistic connection significantly reduces throughput. The Client Core therefore suspends Content Space commits until the termination of the data phase of an opportunistic connection. In the event that a device is reset during an opportunistic connection, then any recent updates in content ownership or metadata are therefore lost. Information pertaining to other nodes can be retrieved during a subsequent connection.

All content items and fragments are stored as files on the device's local storage and/or the removable SD card. In the event of a device reset, updates to the local node's state can be reloaded directly from local storage by iterating through the file system. The state of the local file system is authoritative by definition and replaces any existing Content Space data.

4.3.2 Supported network interfaces

Our BlackBerry prototype implementation includes three concrete implementations of the Connection interface: Bluetooth, Wi-Fi, and an SMS interface (see Chapter 6). The Bluetooth connection class configured to do periodic Bluetooth device inquiry scans. Bluetooth is an ideal radio for neighbour detection due to the low energy cost of a scan [69]. We show in Chapter 7 that the charging habits of most users is sufficient to perform Bluetooth scans continuously without depleting their battery. Upon completion of a scan, the connection object populates a `Bluetooth Neighbour Set` object with `Bluetooth Neighbour` objects containing the Bluetooth MAC address of detected devices, and passes the abstract neighbour set to the Client Core.

After receiving a set of neighbouring devices, the Client Core selects a subset of neighbours whose MAC addresses correspond to trusted Contacts (nodes) in the Contact Store. After selecting this *candidate set* of trusted nodes, the Client Core attempts to connect to a node. Strategies for ranking and selecting nodes is described in Chapter 5. Bluetooth's pairing requirements make it impractical for device-to-device communication. We must therefore use the Wi-Fi interface for communication between devices. Unfortunately, BlackBerry does not support communication in Wi-Fi ad hoc mode [6]. Our current Wi-Fi connection implementation therefore only operates in Wi-Fi infrastructure mode, and

nodes must be associated with a wireless network to exchange large quantities of data directly.

To connect to a neighbouring device over Wi-Fi we require its IP address. Until recently, BlackBerry devices were unable to receive broadcast UDP packets, leaving no mechanism to discover the dynamic IP addresses of older, neighbouring devices. To overcome this problem we created the Bluetooth Lookup Service (BLS). The BLS is an online service that maps that maps a Bluetooth MAC addresses to the LAN and WAN IP addresses of neighbouring devices. We describe the BLS later in this section. We expect that future versions of the system will utilize Wi-Fi Direct, which is available as of version 4.0 of the Android platform [131].

The SMS connection class is a wrapper for an SMS-based transport protocol (SMS-TP). The SMS-TP is stand-alone component of our system, and is described in detail in Chapter 6. Like the implementation of the CTN core, the SMS-TP is also CLDC compliant and communicates with the underlying cellular radio APIs through an `SMS Handler` abstract interface. We have implemented two concrete SMS Handler classes. The first class communicates with the standard BlackBerry native APIs for transmitting SMS messages. The second class is used for experimentation and utilizes a custom Internet-based SMS to transmit messages, which we describe in Section 4.3.4. Unlike the Bluetooth and Wi-Fi connection classes, SMS does not have the concept of a neighbour, nor does it constrain communication to nearby devices. The SMS connection can be used to communicate with any other node at any time. Unfortunately, as we will see in Chapter 6, SMS does not provide sufficient throughput to exchange content. We therefore limit the SMS interface for use only as a control channel. Upon expiration of a control message deadline, the Client Core will schedule the control message for transmission over the SMS interface. Upon successful delivery of a control message over SMS, the message is deleted from the Message Store.

It is important to note that the use of SMS violates the censorship-resistant robustness and unlinkability constraints due to its dependency on cellular network infrastructure. The use of this interface is therefore optional as control messages can alternatively be delivered directly through opportunistic connections. However, given the massive and growing popularity of SMS [78, 145], we believe that the occasional use of SMS to exchange control messages poses minimal risk, especially considering that the messages are encrypted. Our implementation of the SMS-TP can also be instrumented with artificial delays to mimic

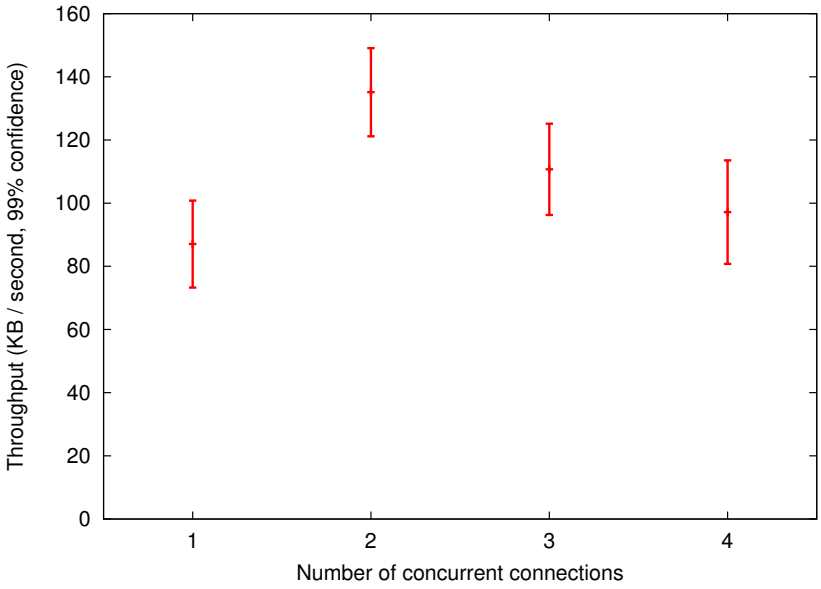


Figure 4.3: Aggregate Wi-Fi throughput vs. number of concurrent Wi-Fi connections.

the SMS transmission behaviour of real-world users [115, 201]. We demonstrate the benefit of using SMS as a control channel in a CTN in Chapter 5.

Each Connection object contains a single thread that is responsible for receiving incoming connections. Connection objects that are capable of detecting neighbouring devices contain a second thread that is responsible for scanning for neighbouring devices. Although the Client Core is thread-safe and can support an arbitrary number of concurrent connections, we only allow two concurrent connections. One connection may be initiated by the thread originating from inside the Bluetooth Connection object, and the Wi-Fi Connection object may received a second incoming connection. While the receive thread is capable of spawning new threads for each incoming connection, our current implementation purposely limits the number of incoming connections to one. This design decision was not arbitrary. We conducted a micro-benchmark to measure aggregate Wi-Fi throughput on a BlackBerry device with respect to the number of concurrent connections. The experiment was conducted using a BlackBerry Bold 9000 device. This experiment was done using a simple BlackBerry application that mimics the I/O characteristics and tit-for-tat behaviour of the protocol described in the next chapter. A 10 MB file was transferred from a laptop to the

BlackBerry device, which spawns a new thread containing a 1 MB receive buffer for every incoming connection. The data is then written to Flash memory. After receiving the data, the application then transmits the contents of a 10 MB file to the laptop. The tit-for-tat process was randomized to ensure that connections were not all sending or all receiving.

Figure 4.3 illustrates the aggregate throughput for the Wi-Fi interface with respect to the number of concurrent connections. Throughput is clearly maximized at two threads. With one connection thread, the time spent blocked on Flash I/O reduced throughput. With two threads, while one thread is blocked the other is receiving data. Interestingly, this frequently causes the operating system's watchdog timer to kill the application due to over utilization of the CPU. With three connection threads we observed that throughput is reduced due to file system contention.

Bluetooth Lookup Service

The BLS is a DNS-like service that maps a device's Bluetooth MAC address to its last known LAN and WAN IP addresses. Like the core and SMS-TP implementations, the BLS is a standalone, CLDC compliant library that can be integrated into any other Java-based mobile application. There are three major components of the BLS: a device-side daemon, the online lookup service, and the lookup API.

The device-side daemon is a continuously running thread that is responsible for monitoring changes to the device's Wi-Fi connectivity. When connectivity is lost, the thread is dormant. When the connectivity is restored or after a user-defined timeout period, the daemon connects to the online lookup service and uploads a base-64 encoded, secure hash of the device's Bluetooth MAC address and its LAN IP address. The use of a secure hash prevents the service from determining the true MAC address of any BLS user if the centralized service is compromised by an attacker. Since MAC addresses uniquely identify devices, and by association their users, it is essential to keep MAC addresses of known CTN nodes private to preserve anonymity.

Upon receiving an update, the service updates the table entry for the given hash string with the given LAN IP address. The service also extracts the WAN IP address from the active socket connection with the client node, and updates the WAN entry for the given hash string. A timestamp associated with each record indicates the freshness of the table

entry. BLS entries expired after 24 hours and are automatically deleted from the BLS database.

Mobile client may query the service for an entry using the base-64 encoded, secure hash of a neighbouring MAC address. If an entry exists, the service will reply with the LAN IP address, the WAN IP address, and a timestamp indicating its age. The mobile client then examines the entry. If the clients share the same WAN IP address, then they are most likely within the same LAN (i.e. both behind the same NAT). In this case, the client should attempt to establish a connection with the LAN IP address. Conversely, if their WAN IP addresses differ than the LAN IP address entry is irrelevant, and the clients should attempt to establish a connection using the WAN IP address.

The use of the BLS in our prototype CTN implementation clearly violates the censorship-resistant unlinkability constraint. The presence of NAT and firewalls also inhibits peer-to-peer connection establishment between peers in different networks. Although this problem could be mitigated by incorporating STUNT [64, 65, 158] functionality into the centralized BLS server, we ignore it because the BLS is only intended to be temporary. As previously mentioned, Wi-Fi Direct is now available on Android smartphones and a wide range of consumer products [195]. Future versions of our system will support these technologies, thus removing the need for the BLS.

4.3.3 User interface

The UI is an essential component of a CTN; however, as a autonomous, continuously running application, the UI is the least-frequently used component. The runtime memory footprint of our implementation reflects this imbalance by maintaining a low memory footprint while attempting to preserve responsiveness to the user. A low memory footprint is trivially achieved by through a strict separation of views from the underlying model. By instantiating UI components when a mobile application is brought to the foreground and subsequently releasing them to the garbage collector when the application is returned to the foreground, the memory footprint of the UI while the application is running in the background is negligible. However, a strict separation between view and model requires elements of the view to be recomputed every time the application is brought to the foreground. This problem manifests itself primarily when viewing the large data structures underlying the Content Store and the Contact Store. The Content Store can easily contain

tens of thousands of content items with various properties and potential sorting characteristics. The Contact Store has comparatively far fewer elements than the Content Store; however, we found it useful to sort this data structure based on nodes' relationship with content and their demands (methods for ranking nodes are discussed in the next chapter). Although sorting is a well-studied problem with efficient solutions, repeatedly sorting these data structures as the user navigates through the UI has a noticeable impact on the responsiveness of the application[1]. Our implementation solves this problem by maintaining pre-sorted lists of objects within the view of each major architectural component. Consistency of these cached view elements is maintained by each architectural component with negligible computational overhead. Each architectural component maintains a reference to the cached view elements using a Java weak reference, thereby allowing the garbage collector to reclaim the objects when memory is needed the device JVM. Any sorted lists within views that are memory resident are updated each time that a new object is updated, added to, or deleted from an underlying data store.

As previously discussed, the resources consumed through wireless I/O and file I/O during opportunistic connection can have a significant impact on the device. While the user is interacting with an application on a CTN-enabled device, it is necessary to preserve a quality user experience. That is, a device should not "lag" as a result of resource intensive tasks being performed by a CTN. Our prototype implementation therefore monitors the BlackBerry idle timer before the most common CPU intensive tasks. The idle timer is reset upon every user gesture, allowing the CTN to reliably determine when the user is actively interacting with the device. We found that introducing a 500 ms delay on socket I/O, computing the hash of content items, and pre-sorting data structures for routing, while the idle timer was less than five seconds, made a significant, positive improvement. The value of five seconds was chosen based on personal interaction with the CTN-enabled device. Periods of non-interaction that are shorter than five seconds in length generally correspond to active interaction with the device. In the absence of this artificial delay, we found that the device was frequently unusable.

[1]The UI latency associated with computationally intensive tasks is not a new problem to mobile application developers. A common design pattern in mobile applications is to perform computationally intensive tasks at the start of the application while the user is presented with a splash screen. Unfortunately, anecdotal evidence derived from previous experiments [143] indicates that users expect a mobile application that is explicitly indicated as active by the device UI to be immediately accessible.

4.3.4 Experimental framework

Our prototype implementation includes an experimental framework that is designed to provide consistent, repeatable experimentation using real mobile devices. Conducting experiments on real mobile devices is essential for capturing the computational and communication constraints present in a realistic deployment. The ability to conduct repeatable, controlled experiments is used extensively to evaluate our routing protocol in Chapter 5.

Our experimental framework has two major components: experiment profiles and a device-side component to download, interpret and manage an experiment. An experiment profile is an XML document that contains aggregate experiment settings, node profiles and individual content and settings, trust relationships, and an event schedule. Experiments are initiated manually through a device-side UI, which retrieves an index of experiment profiles from the web server.

Upon selecting an experiment profile, the experimental framework downloads and parses the profile. Each node profile is uniquely identified by its Bluetooth MAC address. After downloading the experiment profile, each node extracts custom settings from its node profile, as well as a set of URLs of pre-generated content items that it downloads prior to the experiment.

Trust definitions allow nodes to be configured into experiment-specified trust relationships. An experiment profile may contain many trust definitions. Each node extracts only the trust definitions for which it is a member.

Each experiment profile contains an event schedule, which is a timeline of events that occur within an experiment. An event schedule contains many events; however, each event is consumed by only one node. The node, when parsing the profile, discards events unrelated to it. These events include: contact events, demand events, and exit events. A contact event represents a successful Bluetooth scan that identifies one or more other nodes and the duration of emulated collocation. Contact events may be discrete events or repeating.

Demand events represent an expression of demand by a node for a content item with a specified CID. This event is an emulation of a user clicking on a content item in the UI to signal demand for the item. Unlike contact events, demand events can be executed lazily due to the non-determinism in how metadata is propagated through the network at

runtime. Although contact schedules can be defined precisely, metadata propagation is subject to the runtime behaviour of the routing protocol. A user cannot express demand for a content item that is not present in the node's Content Space. If encountered, demand events for unknown items are held until the content items are actually discovered.

Exit events signal the end of an experiment and terminate any recurring events. Consumption of an exit event terminates the experiment and initiates the upload of logs to a web server for subsequent analysis.

After parsing an experiment profile, each node purges all persistent system state from the Contact Store, Content Space, and Message Store. Configuration settings are also reverted to default value. Each node then initializes its Contact Store with its own local identity, its trust relationships, and the identities of the nodes that it is configured to trust. Each device then seeds itself with specified content downloaded from the web server onto the device's local, persistent storage (on BlackBerry, this is typically a removable SD card). The newly downloaded content is immediately added to the Content Space on each device. At the end of the initialization phase, each device is in a known state. Assuming that all nodes are clock synchronized from the PC used to load the CTN software, all nodes begin the experiment at a common specified time.

Our experimentation framework allows for both consistent, repeatable experimentation across a large number of CTN nodes while taking into account the hardware limitations of smartphones. Given the previously discussed limitations of opportunistic communication between resource-constrained devices, preserving hardware limitations is important to maintaining accuracy. Garbage collection, for example, can have a noticeable impact on throughput given that wireless I/O on low-CPU powered devices has been shown to be a CPU-bound operation [128]. This performance bias introduced by experimenting on real smartphone hardware is not represented by existing DTN simulators [35].

Internet-based SMS

SMS is a useful mechanism for exchanging small amounts of control information between devices. However, providing cellular network connectivity to every device in a large, 55-node experiment would be expensive. We developed an Internet-based SMS to emulate SMS service in the absence of cellular network connectivity. The Internet-based service consists of two components: an SMS Handler and an Internet-based SMSC. The SMS

Handler architecturally sits below the SMS-TP stack and is responsible for sending and receiving 140-byte SMS messages to/from the SMS-TP.

The Internet-based SMS stack addresses nodes by the string representation of their Bluetooth MAC address. Upon receiving an address-message pair, the SMS Handler component uploads the message to an SMSC component running on an Internet server. The messages are stored in a persistent database, but expire if not delivered within 24 hours. Periodically, a receiving thread within each SMS Handler queries the server with its Bluetooth MAC address checking for new messages. The server responds with all pending messages destined for the given MAC address. Once a message is successfully delivered, it is removed from the SMSC's database.

4.4 Chapter summary

In this chapter we have outlined the software architecture and implementation details of our prototype CTN system. To satisfy the CTN criteria, our architecture is incrementally deployable and fully decentralized, but may optionally, and/or intermittently, utilize infrastructure to exchange SMS messages for control purposes.

We have described how trust is established between nodes through direct pairwise social interaction where identities are exchanged over short-range, secure wireless links and never shared with non-trusted nodes. All communication between CTN nodes takes place within the trust topology formed through these pairwise trust relationships. Our architecture leverages the structure of the trust topology to create a simple, efficient scheme for organizing and explicitly expressing demand for content contained within the CTN.

Our architecture purposely omits one major component: the protocol governing communication between two mutually trusted devices during an opportunistic connection. The next chapter explores this topic in greater detail.

Chapter 5

A Laissez-faire Approach to DTN Routing

5.1 Introduction

The decentralized formation of trust relationships between nodes provides both a secure means of exchanging identities and communication privacy. The merging of content metadata into each node's Content Space provides a means for nodes to unambiguously express their demand for content within the network. We now build upon this functionality, and describe a protocol for pairwise communication during an opportunistic connection that implements routing. As its name suggests, Laissez-faire is a *hands-off* approach to DTN routing. Instead of routing content from content holders to nodes that demand the content, Laissez-faire creates a free market for content between nodes. Under Laissez-faire, nodes accumulate debt by obtaining content from other nodes. Conversely, nodes allocate credit by giving content to, or carrying content for, other nodes. The goal of retrieving content for others is to accumulate credit, so that future demand for content can be retrieved on the node's behalf in payment of debt. Under Laissez-faire, all nodes are assumed to operate selfishly to satisfy their own demands.

Content exchange between two nodes during an opportunistic connection is a tit-for-tat process. Each node takes turns requesting a fragment of a content item from the other. When selecting fragments from a neighbouring node, we assume that nodes always prefer to retrieve fragments that satisfy their own demand over fragments of content demanded by other nodes. That is, satisfying a node's own demands is always more important than reducing debt and satisfying the demands of others. Although this strategy may not be optimal, it is an assumption that we believe is reasonable to make.

This chapter makes two contributions:

- We propose the *Laissez-faire* framework, a hands-off, incentivized approach to CTN routing. We believe that the optimism regarding node participation in existing DTN-based mobile systems is incompatible with our goal of releasing the work as open source software. As open source software, anyone in the world is free to modify the code and manipulate the underlying routing protocol to their own advantage. No such protocol is defined in Laissez-faire. Instead, nodes are only responsible for maintaining their own records of content exchange, and to implement the operations of a pairwise tit-for-tat protocol. Yet our approach still allows for end-to-end delivery of content. We believe that the simplicity of Laissez-faire can serve as the foundation for future DTNs targeted for average smartphone users.

- We implement the Laissez-faire framework within the prototype CTN system described in the previous chapter, and evaluate Laissez-faire across a cluster of smartphones using our experimentation framework and through a three-week, 36-participant field trial. Although Laissez-faire does not *mandate* any specific routing behaviour, we propose several practical strategies for retrieving content fragments. We demonstrate the trade-offs between strategies under varying controlled and real-life network conditions.

  We introduce three simple strategies for ranking the demands of other nodes. Each strategy is based upon minimizing either a node's demand or debt relationship with other nodes, or the buffer allocation cost of carrying content fragments to nodes. Once nodes have been ranked, we select fragments for nodes using a fragment selection strategy. We introduce four sample fragment selection strategies. Fragment selection strategies are also used to eject fragments from the storage buffer when the buffer reaches capacity. We compare both the node ranking strategies and fragment selection strategies with a naïve approach that simple selects nodes and fragments uniformly randomly.

This chapter begins with an overview of the protocol operations that may take place during a tit-for-tat exchange. In Section 5.3 we introduce several practical mechanisms for ranking fragments and the demands of other nodes in the CTN. In Section 5.4 we conduct a preliminary evaluation under controlled network conditions using our sample strategies for ranking nodes and selecting fragments. These experiments are followed by a field trial, which we describe, and present the results of, in Section 5.5.

5.2 Protocol operations

During an opportunistic connection between nodes n_i and n_j, each node takes a turn executing one of the operations detailed in the following subsections (in any order) or pass their turn. However, the following ordering is used in our prototype implementation. When both nodes have elected to pass their turn, then the tit-for-tat protocol is terminated and the nodes disconnect from each other. Each operation is described from the perspective of n_i.

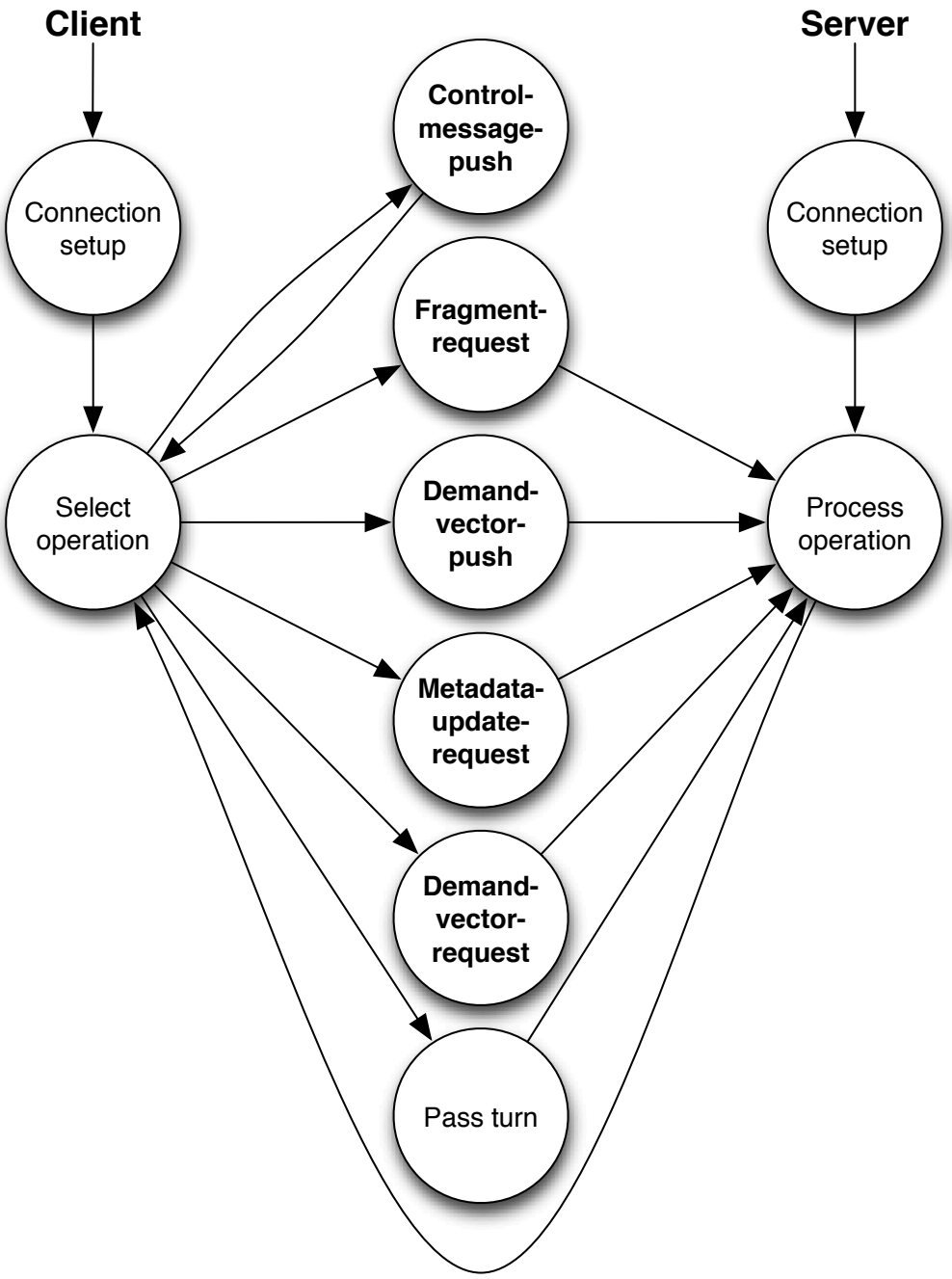


Figure 5.1: Laissez-faire protocol operations.

The protocol operations and the transitions between them are also illustrated in Figure 5.1. Although nodes operate as peers, we denote the client as the node that initiates the connection and the server as the node that accept the connection.

5.2.1 Control-Message-Push

As we have previously discussed, control messages are used to maintain and propagate the state of the system to other nodes. Text messages, similar to text messages conventionally sent over SMS, are also encapsulated as control messages. Text messages allow users to take advantage of the secure communication architecture underlying the CTN to exchange personal messages. Control messages are accompanied by a delivery deadline, have negligible communication overhead, and may be assumed to always benefit the recipient. These messages therefore have the highest priority and do not count against a node's turn in the tit-for-tat protocol. If pushing a control message were to count against a tit-for-tat turn, then nodes, in the service of their own demands, would be better off spending their turn retrieving a fragment than passing a small control message.

5.2.2 Fragment-Request

Retrieving fragments is the highest priority operation after exchanging control messages. During this operation, a node may retrieve a fragment of content for itself or on behalf of another node. Underlying the fragment exchange is a pairwise, byte-based accounting scheme. Each byte transferred from n_j to n_i is deducted by n_j from n_i's current balance, until n_i's credit limit is reached. We refer to this quantity as the *debt threshold*. Enforcement of the debt threshold is dependent on a node's sensitivity to resource costs. Recall from Chapter 1 that content activists are insensitive to resource costs, and are more concerned with sharing censored content than receiving a reciprocal benefit. For nodes that are sensitive to resource costs, fragment requests that meet or exceed the debt threshold will be rejected, and n_i is notified that it has reached its credit limit.

We assume that nodes have a large, diverse, insatiable demand for content, whose aggregate size greatly exceeds link capacities and devices' storage. This assumption represents the worst case scenario and is compatible with existing data on music and video consumption behaviour [59, 98]. Therefore, nodes must strategically select fragments to

maximize their own utility. We further assume that n_i always retrieves its own fragments before considering the demands of other nodes.

After retrieving its own demanded fragments, n_i may consider the demands of other nodes. Nodes that have reached their debt threshold are excluded from consideration. When multiple other nodes demand content available at n_j, n_i must prefer to select fragments for one node over another. Retrieving fragments on behalf of other nodes has a cost. Fragments consume storage capacity on the device, potentially causing *less valuable* fragments to be discarded. They also increase n_i's debt position with n_j, consume a turn in the tit-for-tat exchange, and consume energy during their retrieval. Fragments for other nodes must therefore also be retrieved strategically to maximize n_i's utility. We refer to these strategies for ranking the demands of other nodes as *node ranking strategies*, which we explore in Section 5.3.1. Once a node is selected as the most 'valuable' node, we apply the fragment selection strategy of n_i to select fragments. We explore strategies for selecting fragments in Section 5.3.2.

If a requested fragment is no longer available, the request must therefore be rejected. Node n_i may then perform another operation. A rejection caused by meeting or exceeding a credit limit signals the end of the fragment request operation, since all subsequent fragment requests will also be rejected. In general, nodes' debt thresholds may expand or contract over time depending on the degree of resource scarcity or as a node's sensitivity to resources changes. However, this work assumes that debt thresholds are static.

5.2.3 Demand-Vector-Push

If no fragments are available, the next logical operation is to transmit a demand vector to n_j to allow n_j to retrieve fragments for n_i from other nodes. In our current implementation, demand vectors from nodes outside of a trust relationship (i.e. nodes that are 2 "hops" away from n_j), are unioned into a single demand vector by n_i. This new demand vector is then appended to the demand vector pushed by n_i to n_j.

After receiving a series of demand vectors, n_j requests metadata corresponding to CIDs of any unknown content. Retrieving metadata allows n_j to discover new content without the cost of a turn, and facilitates n_j's future retrieval of the content.

5.2.4 Metadata-Update-Request

The Content Space is an integral component of a CTN. The metadata update request from n_i to n_j retrieves the current Content Space known to n_j. The state includes both the complete and partial content held by the trusted node n_j, and the complete content held by other nodes within the CTN. During a metadata update request, n_j transmits two hash values to n_i. The first value corresponds to all content, both partial and complete, that currently exists on n_j. The second value corresponds to all complete content that exists on nodes outside of the trust relationship that are reachable by n_j. We refer to the latter as the *reachable set*. If the hash values match n_i's values, then both nodes have the same view of the Content Space and the operation is complete. If the first hash fails then n_j discloses the all of the CIDs of the content on the device. As previously discussed in Section 4.1.3, n_i would subsequently update the timestamp of each content item associated with n_j with the current timestamp. Node n_i would then update its hash for n_j with the hash of the newly received set of CIDs.

If the hash of the reachable set fails, then n_j must synchronize n_i's view of CS_k for each node n_k that is reachable by n_j. The synchronization process begins with n_j disclosing a series of tuples that represent the content known to exist on each reachable node. Each tuple contains the following data: $\langle timestamp, GUID_k, hash(cs_k)\rangle$ for each node n_k, where cs_k is the content known to exist on n_k. Each timestamp corresponds to n_j's previous encounter with n_k. If the hash differs then n_j should disclose cs_k to n_i using the process described above. However, if n_k is trusted by n_i, then the timestamp should be examined to determine which set of metadata is the most current. If n_i's last contact time with n_k is earlier than n_j's last encounter, then n_j transmits the complete state of cs_k to replace n_i's stale state. Otherwise, if n_k is not trusted by n_i, then the timestamp is ignored by n_i.

5.2.5 Demand-Vector-Request

In the event that all previous conditions are met, n_i may solicit a fresh copy of n_j's demand vector. Upon request, the node n_j responds with its current demand vector.

Each node maintains a record of previously performed operations to minimize redundant repeated operations. If a node fails to participate in the tit-for-tat protocol, its malfeasance

can be punished by simply distrusting the node. We don't consider this to be an issue given the real-world trust relationship that underlines pairwise interaction in a CTN.

One key design requirement emerges from the Laissez-faire framework: the need to rank both nodes and fragments.

5.3 Fragment selection

The Laissez-faire framework does not mandate any specific fragment selection mechanism. As an open system, nodes are free to allocate and expend resources in an attempt to maximize their own benefit. However, we assume that nodes prefer to retrieve their own fragments over the fragments demanded by other nodes. In the absence of locally demanded fragments, nodes should select fragments on behalf of other nodes.

In this section we first describe strategies for ranking other nodes, followed by strategies for retrieving fragments on behalf of other nodes.

5.3.1 Node ranking strategies

With many nodes demanding content, we require a mechanism at each node to rank the demands of one node over another and subsequently select a fragment. We present three simple strategies for ranking nodes. Each strategy is based upon one of three major properties within a CTN: *demand* for content, *debt* accumulated in satisfying demand, and the physical space or *buffer* capacity allocated to storing content fragments.

The three node ranking strategies are:

- **Decreasing demand**: Through the Content Space, each node maintains a record of the last known content (both partial and complete) present on other nodes in the CTN. The demand strategy ranks a trusted node in decreasing order of the amount of demanded content present at that node. Nodes that possess more of n_i's demanded content are favoured over nodes that possess less demanded content. Once the nodes are ranked, the highest ranked node is selected. Fragments for that node are then selected by n_i using n_i's fragment selection strategy.

- **Decreasing debt**: Recall that every node maintains its debt with every other trusted node as well as an account of every other node's debt to it. It also has a debt threshold, the upper bound on the amount of debt another node may owe it. If, during the process of n_i satisfying a node n_j's demand, the debt threshold is met, n_j must reciprocate with content transfer to n_i to reduce its debt before subsequent content may be transferred. Delayed repayment of debt has the consequence of introducing delay between two nodes that could otherwise be exchanging content, it may also diminish the node's perceived value during third party transactions. For example, node n_i may be unwilling to retrieve content on behalf of node n_j from n_k if n_j's has reached n_i's debt threshold. The debt strategy is designed to proactively reduce n_i's debt position with other nodes by favouring nodes with the greatest debt position. Like the demand strategy, once a node n_i selects the top ranked node n_j using this strategy, fragments for n_j are selected by n_i using n_i's fragment selection strategy.

- **Buffer**: Neither the demand or debt strategies consider buffer displacement, which is the storage cost of transporting fragments to a node. The buffer strategy is designed to minimize buffer displacement by favouring nodes with the lowest expected displacement cost of carrying demanded fragments to it. This cost of carrying a fragment to a node is amortized over all nodes that demand the fragment.

  We denote $\hat{c_j}$ and c_j as the exponentially weighted moving average inter-contact time and last contact time with node n_j. The expected duration until the next contact with n_j is computed as follows: $\max(0, \hat{c_j} - (now - c_j))$. We also denote d_j as the demand vector of n_j. The buffer displacement cost of a fragment m_i amortized over all nodes that demand the fragment is computed as follows:

$$fragment\_size(m_i) \left(\frac{\max_{n_j | m_i \in d_j} (\hat{c_j} - (now - c_j))}{|n_j | m_i \in d_j|} \right) \tag{5.1}$$

  In this formula, the $fragment\_size(m_i)$ is the size of fragment m_i. The second portion of the formula first examines the demand vector d_j of each node n_j who is trusted by n_i. For each node n_j that demands the fragment m_i, we examine n_i's inter-contact time with n_j and the last time that n_j was encountered. We then select the node from this set that is expected to be contacted last. The duration that m_i

must be stored for delivery to the selected node is the maximum buffer displacement duration for m_i. We then amortize the buffer displacement duration by the number of nodes that demand m_i.

The displacement cost excludes nodes whose debt threshold has been reached. Unlike the debt strategy, where the value of fragments can be determined prior to a connection, the buffer strategy is time dependent. Displacement costs must be calculated during a connection based upon the neighbouring node's available fragments.

These strategies are also used to discard fragments when the device's storage buffer has reached capacity. Fragments destined for lower ranked nodes are deleted first.

5.3.2 Fragment selection strategy

During an opportunistic connection, a node may simultaneously demand many fragments. Existing systems approach this problem by assigning a uniform value to all desirable fragments [66, 86, 144]. Having equal value, fragments are retrieved at random until the content item is completed. However, this approach is only practical for light workloads and breaks down as content demand exceeds link capacity. The primary goal of all nodes is to retrieve the fragments necessary to satisfy demand. Given that CTNs operate as disconnected nodes, we assume that demand always exceeds link capacity and that nodes must rank fragments.

We introduce the concept of a fragment selection strategy to govern the process of selecting a fragment from a neighbouring node. The following sample strategies form the basis for subsequent evaluation in Section 5.4 and Section 5.5. However, it is important to emphasize that, like node ranking strategies, nodes are free to implement any fragment selection strategy that they choose.

- **Least data remaining first**: This strategy favours fragments of a content item with the fewest bytes remaining. This selection strategy has the side effect of favouring smaller content over larger content. For example, a small image at 0% completion is favoured over a large video file at 90% completion.

- **Greatest data remaining first**: Alternatively, a node may prefer to make progress on content items with the greatest outstanding amount of data. This strategy prefers fragments of large content to fragments of smaller content. Each received fragment reduces the quantity of data remaining, thus reducing the value of subsequent fragments for that content item.

- **Least fragments remaining**: This strategy prefers fragments of a content item with the fewest remaining fragments. Unlike the least and greatest-data-remaining-first strategy, this approach is agnostic to the size of the content.

- **FIFO**: This strategy prioritizes fragments in order of request time. The longer the content has been demanded, the higher priority the fragment is. In our prototype implementation, this simply corresponds to the position of a CID in a node's demand vector. This strategy is therefore agnostic to both the content size and completion status.

  In practice we anticipate that the FIFO strategy would be the most popular since it is the most intuitive and closely resemble existing content acquisition systems, such as BitTorrent[1].

5.4 Evaluation under controlled conditions

We conduct a series of experiments to evaluate Laissez-faire under controlled network conditions. As no prior work exists to infer how disconnected, mobile users would (a) form trusted relationships with other nodes, (b) demand content from other nodes in a disconnected network, or (c) contact other trusted nodes, this state of our evaluation is designed to illustrate the intrinsic properties of different node ranking and fragment selection strategies. We extend our analysis through a field trial in Section 5.5.

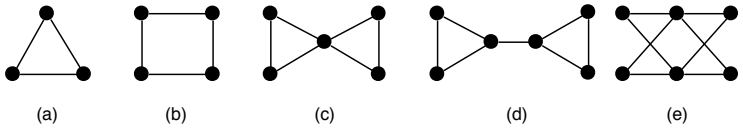


Figure 5.2: Experiment topologies

5.4.1 Methodology

Our evaluation takes place over five simple communication topologies in increase order of complexity. These topologies are illustrated in Figure 5.2. Each edge in the graph indicates a potential communication path between nodes. In each experiment the communication graph is a time-varying multigraph; edges are activated and deactivated at specified intervals for a fixed 90 second connection duration. Given that human mobility is highly diverse [61], this is clearly not a realistic contact schedule. We chose this schedule to isolate the influence of human mobility on our study. Since our communication topology is static and nodes are assumed to share metadata honestly, the trust topology has no impact on our evaluation. All nodes are configured to be mutually trusted. Each experiment was repeated ten times.

We conduct our experiments using BlackBerry Bold 9000 devices. Each device is equipped with a 8 GB micro SD card and we seed it with 1 GB of randomly generated unique content, ranging from 1 MB to 16 MB in size. Each device is configured with a debt threshold of 128 MB and 256 MB of storage is allocated to storing fragments demanded by other nodes. We also assume that once a node expresses demand for a content item, it is never retracted. That is, items are never removed from a demand vector unless they have been retrieved.

Metrics

This evaluation focuses on three key metrics: the *delivery ratio*, the *delay*, and the *fragment drop ratio*. The delivery ratio is the proportion of node content demand that is satisfied over the course of a trial. The delay is the mean duration between the user signaling

[1]Most BitTorrent clients schedule downloads by FIFO unless the user has manually adjusted the priority of a torrent.

demand for an item and having the demand satisfied. These metrics are analogous to the delivery ratio and delay metrics in existing DTN routing protocol evaluation [80]. The drop ratio is the proportion of fragments retrieved by a node that are discarded either because the node demanding the fragment has received the demanded content item through some other path and no longer needs the fragment or if the node's storage buffer has reached capacity and fragment(s) must be dropped. The drop ratio therefore captures the degree of wasted communication in the system. Fragments that are deleted after having been transferred to another node are not considered a wasted acquisition. These fragments are therefore not counted as dropped.

Discussion on metrics

Although our metrics for evaluating CTNs are compatible with existing DTN metrics, they can be biased by both our CTN communication requirements and the topology of the network. Nodes in a CTN may only demand content that has previously been discovered through a trusted node. Although this is an expected result that follows from not communicating with non-trusted nodes, it has the effect of narrowing the set of content that is exposed to a node. Both delivery ratio and delay are positively biased by the fact that nodes may only demand content contained on, or through, a trusted node, rather than any arbitrary node in the network. A node cannot, for example, demand a highly popular content item unless it can be obtained through the nodes that it trusts. Moreover, in a real-world deployment, it would not be surprising if mutually trusted nodes encountered each other, on average, more frequently than nodes that are two or more hops apart in the trust topology.

The positive bias is strongest in the degenerate CTN, a CTN consisting of two mutually trusted nodes. In this network, assuming that the two nodes contact each other, the delivery ratio will be convergent to 1.0 and the delay will be purely influenced by the inter-contact time of the two nodes. We examine the degenerate case in Section 5.5.2.

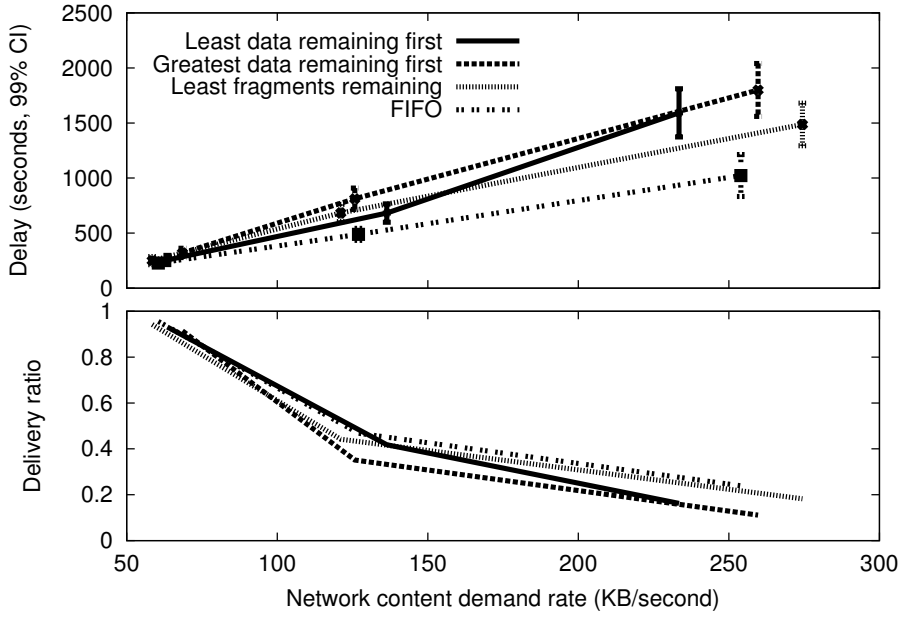


Figure 5.3: Comparison of fragment selection strategy for a three-node topology.

5.4.2 Effect of fragment selection strategy

Experiment 1

We conduct a simple experiment to compare the differences between fragment selection strategies discussed in Section 5.3.2. To isolate the effect of different node ranking strategies, we limit this experiment to the simple three-node topology. In a three-node topology, a node either retrieves its own fragments or fragments for the single node not being communicated with. We also introduce an additional fragment selection strategy: *Random*. The random fragment selection strategy selects (and discards when the buffer capacity is reached) a uniformly random fragment. The random strategy is analogous to the approach used in existing systems. All nodes in this and the following experiments are assumed to be sensitive to resource costs.

In each trial, edges in the triangle communication topology are selected uniformly randomly and enabled for 90 seconds. Upon termination of a connection, the node that initiated the connection uniformly randomly selects a varying number of known content

items from another node in the network. The number of content items selected is chosen to yield a specific content demand rate. Figure 5.3 illustrates the delay and delivery ratio for each fragment selection strategy vs. the aggregate rate of new content demand in the network.

Not surprisingly, a FIFO ordering outperforms all other methods in terms of both delay and delivery ratio because it is designed to greedily minimize this property. The least-fragments-remaining strategy behaves similar to FIFO for low rates of demand, but degrades as demand exceeds the communication capacity of the network. Once the link is saturated demand at each node accumulates. When faced with multiple new demands, this strategy selects one at random, independent of its age. The remaining two strategies starve either large (least-remaining) or small (greatest-remaining) content demands and have comparatively low delivery ratios. Demanded content that is retrieved arrives after a significantly longer period than the other strategies. Fragment duplication in this experiment is approximately zero since nodes nearly always demand content from the neighbouring node. The random fragment selection strategy barely functions when demand exceeds the capacity of the network. By uniformly randomly selecting (and discarding) fragments, we were only able to deliver a few content items over the entire experiment with very high delay. The poor performance of the random strategy was omitted from Figure 5.3. Although at times user-preference may supersede efficiency, favouring, for example, smaller audio files over larger video files, the remainder of this stage of our evaluation will be based upon the FIFO fragment selection strategy.

5.4.3 Effect of node ranking strategies

We conduct four independent experiments to illustrate the properties of each node ranking strategy.

Experiment 2: Demand, debt, and buffer homogeneity

Our first experiment is designed to evaluate the behaviour of each node ranking strategy under homogeneous conditions. Our procedure for this experiment is identical to Experiment 1. As in the previous experiment we introduce a naïve, *Random* strategy for

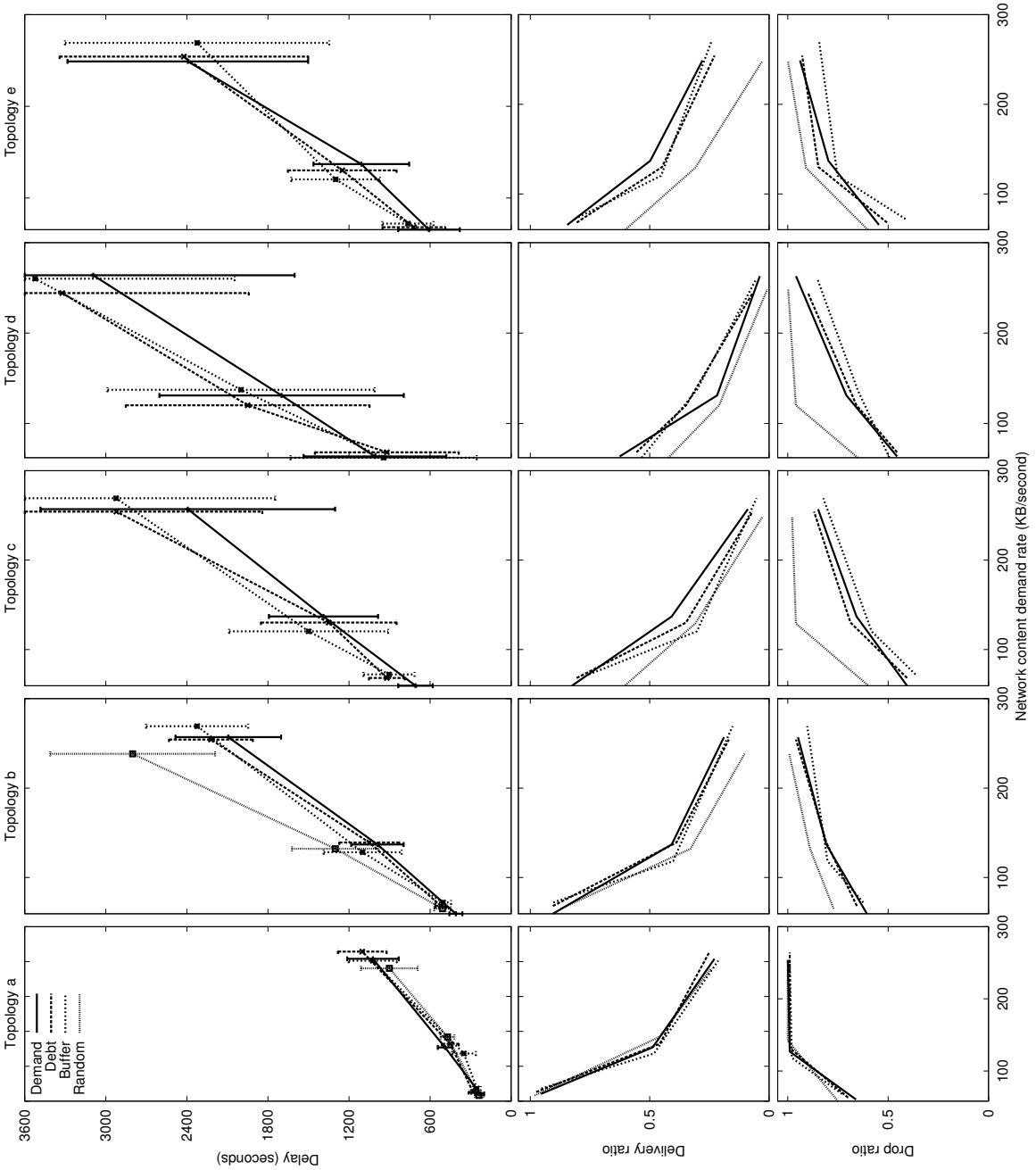


Figure 5.4: Analysis of node ranking strategies.

comparison. However, instead of selecting a fragment, the random node ranking strategy selects a uniformly random node.

Figure 5.4 illustrates the delay and delivery ratio with respect to the mean content demand rate over each topology. Over our simplest, most connected communication topology (a), the choice of strategy has little effect on the system; nodes are almost always able to retrieve their own content directly. As aggregate demand increases, nodes predominantly retrieve their own content (that they do not drop). However, content that is retrieved is almost never delivered since the destination node has almost always retrieved the fragment themselves. This results in an increase in the drop ratio.

A node's closeness centrality is the sum of the distances to all other nodes. Thus, the more central a node is, the lower its total distance to all other nodes. As node centrality decreases, we observe that strategy begins to have an effect. At low rates of demand they are statistically equivalent; however, even in a four-node topology we see that the random strategies' indiscriminate selection and ejection of fragments results in both an increased delay and drop ratio as well as a substantial decrease in delivery ratio. We omit the random strategy from the delay graphs for topologies in c, d, and e in Figure 5.4 to avoid its poor performance being misinterpreted good performance. Unlike the other three strategies, the random strategy is unable to transport content across two hops when aggregate demand exceeds the network's capacity. All content successfully retrieved using the random strategy is retrieved only from adjacent nodes, typically at the start of the experiment before demand has accumulated. This results in an artificially low mean delay.

As the aggregate rate of demand increases, we observe that the demand strategy outperforms the others in terms of delay. By retrieving fragments on behalf of nodes holding demanded content, this strategy has the tendency to accumulate content, which can be exchanged with the node during the next encounter without significantly affecting the nodes' debt relationship. However, these results are not statistically significant. Overall, we also observe a reduced fragment drop ratio under the buffer strategy. This result is primarily due to the strategy's greedy approach to reclaiming buffer capacity: larger fragments are always discarded before smaller ones. The space reclaimed through the deletion of large content items can often accommodate several new items.

Topologies c and d introduce bottlenecks into the path, causing a significant decrease in the delivery ratio and a corresponding sharp increase in drop ratios. The bottleneck further serves to amplify delay during heavy content demand levels: mean delay approaches an

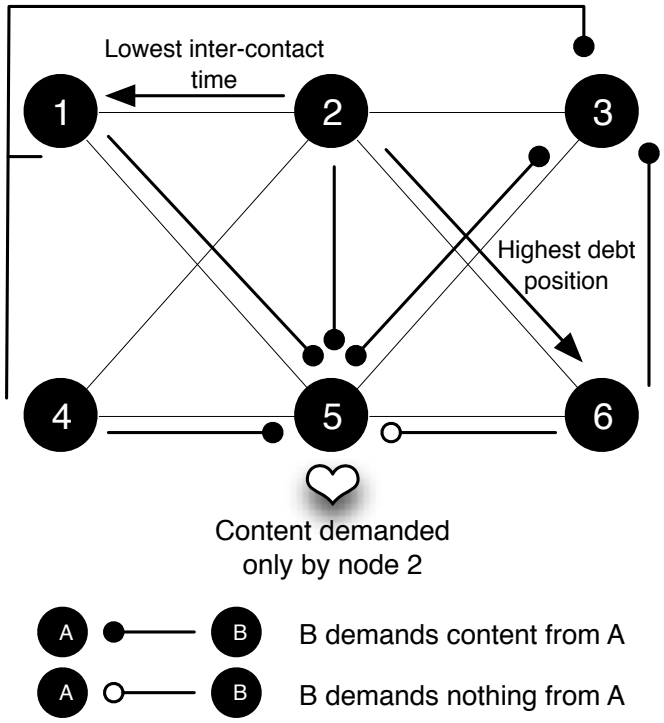


Figure 5.5: Experiment 3 (demand heterogeneity) configuration.

hour, introducing significant disparity in delay between 1 hop away and content on the other side of the bottleneck(s). In our final topology, we decrease node centrality by introducing additional edges in the contact graph. All nodes are reachable by two disjoint paths of at most two hops. At low demand rates, nodes in this topology experience delays and delivery ratios comparable to those in topology b. However, we observe a significant increase in the variability largely due the corresponding increase in drop ratio. Surprisingly, both the delivery ratio and drop ratio increases in topology e compared to topology d. This result is due to the increase in the number of paths in the network. While more content may be transferred, there is a greater probability that fragments are retrieved from another node.

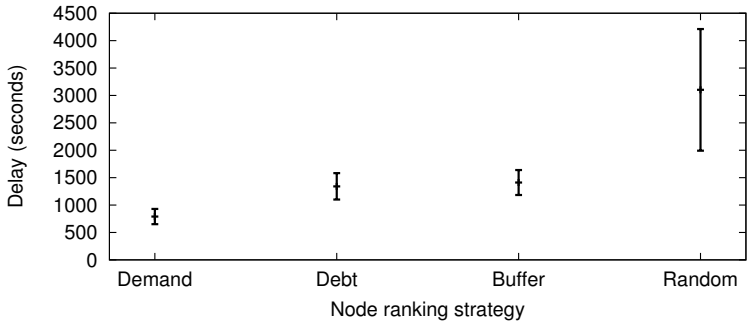
Strategically selecting fragments based upon node's unique demands, debt, and buffer capacity seems to outperform the naïve, random approach used by existing systems. However, under varying degrees of demand, all three strategies are statistically equivalent at a 99% confidence level. Our final three experiments are designed to highlight the conditions where each strategy excels.

Experiment 3: Demand heterogeneity

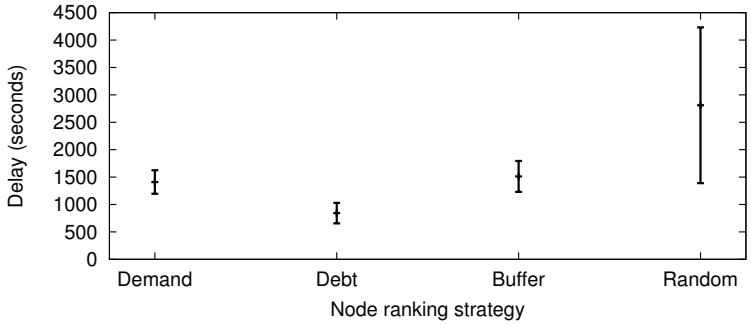
Our final three experiments are based upon topology e in Figure 5.4. Each experiment is designed to favour the use of a specific node ranking strategy. All nodes are configured to select fragments using the FIFO fragment selection strategy. In all three experiments, node 2, as illustrated in Figure 5.5, demands ten random content items present on node 5, the only node not directly connected to it. In all three experiments, there are four disjoint paths between the two nodes.

The intermediate nodes demand content from specific nodes as indicated in Figure 5.5. Figure 5.5 also illustrates the debt relationship that exists between nodes in the experiment topology. All nodes except node 6 have significant demand for content on node 5, and would therefore retrieve their own fragments first. Node 6 will therefore always retrieve content from node 5 on behalf of one of the other four nodes in the experiment. The rate of demand is approximately twice the link capacity of the network. Node 2 is the only node that demands nothing from node 3, and is therefore going to select content only for other nodes during connections with that node. The contact schedule is uniformly random as in the previous experiment; however, node 2 has been modified to reduce its inter-contact time with node 1 to half the value of any other node. This modification to node 2 causes node 1 to always be favoured under the buffer node ranking strategy. The experiment terminates when node 2 receives all of its content, which results in a constant delivery ratio of 1.0 at node 2.

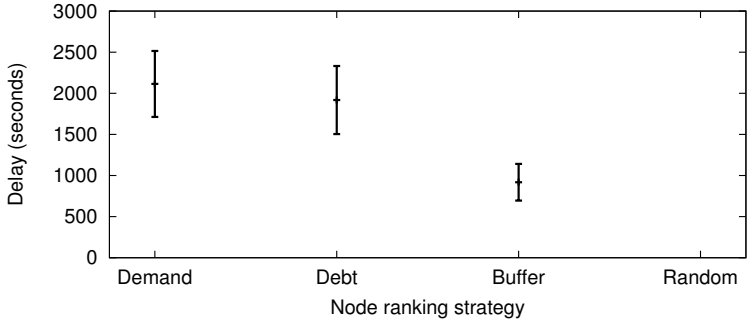
Recall that goal of each node is to obtain content with the least delay. Figure 5.6(a) illustrates our results from the perspective of node 2. Delay is clearly lowest using the demand node ranking strategy. Using the demand ranking strategy, node 6 initially favours content that is demanded by node 3 since node 3 is the only node that possesses content demanded by node 6. However, once node 2 begins to retrieve fragments from node 3, it will be preferred by node 6 over the other remaining nodes. Subsequently, an increased amount of content will be routed from node 5 to node 2 via node 6. Given the poor performance of the random node ranking strategy did not result in a delivery ratio of 1.0, under this strategy we terminate trials when a single demanded item is delivered to node 2.



(a) Mean delay in delivering content to node 2 in experiment 3 (heterogeneous demand).



(b) Mean delay in delivering content to node 2 in experiment 4 (heterogeneous debt).



(c) Mean delay in delivering content to node 2 in experiment 5 (heterogeneous inter-contact time and constrained buffers).

Figure 5.6: Delay for each heterogeneous configuration.

Experiment 4: Debt heterogeneity

We now create a scenario designed to favour the debt node ranking strategy. In this experiment, each node's demand for content is uniformly distributed over the network. Nodes demand content at a rate that is approximately the same as the link capacity of the network. However, nodes 1, 3, 4, and 6 are initialized to have reached their debt threshold with node 5. They will not be able to retrieve any content from node 5 until they have reduced their debt position. Moreover, these nodes will not retrieve any content for node 2 until they have both reduced their debt and satisfied their own demand. The content demands of node 5 therefore have a strong influence on the system. All other debt positions are initially neutral.

Figure 5.6(b) illustrates the results of this experiment. By collecting content on behalf of node 5, each node increases the quantity of content that may be retrieved by its other neighbours who all rank the demands of node 5 highest under the debt node ranking strategy; all nodes need content demanded by node 5 to reduce their debt position with the node. Although the demand and buffer node ranking strategies also result in the successful completion of the experiment, they do not perform as well as the debt node ranking strategy because they are inhibited by initial debt in the system. The debt node ranking strategy has the effect of proactively pushing content towards node 5. As in the previous experiment, the random node ranking strategy performs poorly, and the experiment terminates once a single content item is delivered to node 2.

Experiment 5: Contact heterogeneity with constrained buffers

Our final experiment is designed to provide conditions where the buffer strategy excels. In this experiment, we reduce each node's buffer capacity from 256 MB to 16 MB, enough to hold a single copy of the largest content items in our system. In this experiment, we artificially reduce the calculated inter-contact time between nodes 1 and 2 to create a favourable path through node 1 between nodes 2 and 5. That is, when calculating the buffer displacement cost of carrying content fragments between nodes 1 and 2 using the buffer node ranking strategy, both nodes artificially reduces the weight by half. However, it is important to note that the true contact schedule between nodes and the subsequent inter-contact time is unchanged in this experiment. Edges in the contact graph (topology

e) are activated uniformly randomly as in the previous experiments. Thus this experiment does bias the true delay in delivering content to node 2.

Figure 5.6(c) illustrates the results of this experiment. During each connection between nodes 1 and 5, node 1 preferred to retrieve content for node 2 over any other node in the system due to the comparatively lower storage cost. During connections between nodes 1 and 2, node 2 immediately retrieves content fragments that had been retrieved on its behalf. Similarly, node 2, when connected to nodes 3, 4, or 6 prefers to retrieve content for node 1, again because of the comparatively lower storage cost. Under the heavy buffer constraints of this experiment, no content could be delivered under the random strategy.

Each of the three experiments was designed to favour a specific node ranking strategy. The results should not be surprising. In Section 5.5 we re-evaluate the performance of the three node ranking strategies through a field trial.

5.4.4 Robustness of Laissez-faire to free-riding nodes

Participation in a CTN requires that nodes allocate and expend resources for other nodes. However, there may exist free-riding nodes that are unwilling or unable to share resources. We conduct a final simple experiment to assess the impact of free-riders on delay and delivery ratios. We assume that once a node reaches its debt threshold then no further fragments will be delivered to the node. As in the previous experiments, we continue to assume that all nodes are sensitive to resource costs. If all nodes are content activists and insensitive to resource expenditure, then free-riders have no impact on the system. This experiment uses the homogeneous conditions outlined in Experiment 2. The experiment terminated after one hour. We reduce the debt threshold to 64 MB to exaggerate the effect of free-riding and configure nodes to randomly select one of the three node ranking strategies. As in the previous experiments, we configure each node to use the FIFO fragment selection strategy. Only one content item is requested after each connection to produce a mean aggregate demand rate of approximately 64 KB/s.

Given that free-riding nodes will be cut-off by resource-sensitive nodes once their debt threshold is reached, the result of this experiment may be easily predicted. However, we conduct 10 trials. In each trial, we uniformly randomly select a single node to be the free-rider. Free-riding nodes may be trivially introduced into the system by configuring

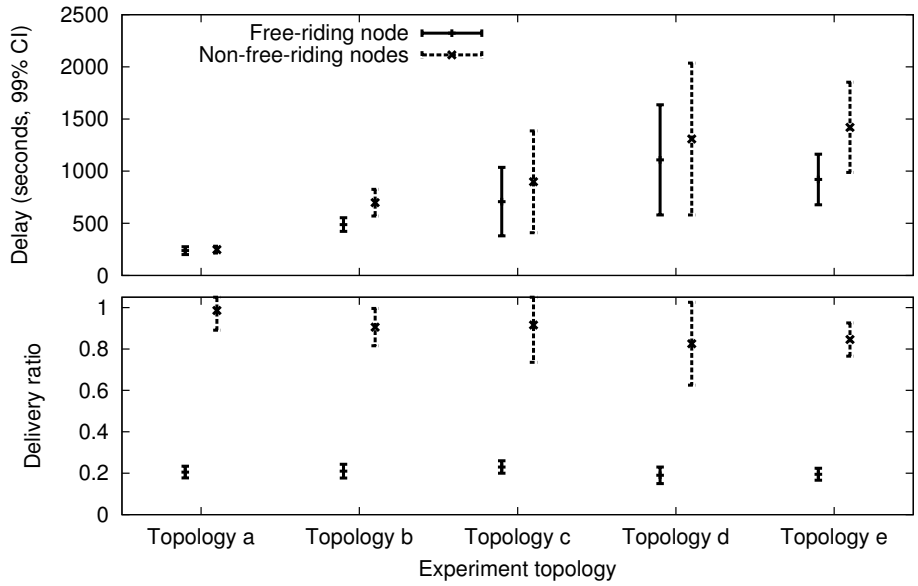


Figure 5.7: Delay and delivery ratio for free-riding and non-free-riding nodes.

them to never disclose metadata. Nodes that are known not to have any content after a metadata update has been performed are never solicited for a fragment. However, these nodes collect metadata and request content at every opportunity.

The delay observed for free-riding nodes are comparable to our previous observations. However, free-riding nodes had significantly lower mean delivery ratios: approximately 20% compared to the near 90% mean delivery ratio of non-free-riding nodes. Once a free-rider has met the debt threshold of their neighbouring nodes, subsequent requests for content fragments will be rejected. Figure 5.7 illustrates the delay and delivery ratios for both free-riding and non-free-riding nodes for each topology.

The delay and delivery ratio for non-free-riding nodes is unchanged for topology a. This result is not surprising since the introduction of a free-rider in this simple topology does not increase the average path length between the other pair of nodes.

Non-free-riding nodes experienced much higher delays under topologies b and e than their free-riding counterpart. This result is primarily due the decreased number of paths between the content source and destination. In the case of topology e, the effect is greatest when either of the central, four-degree nodes is selected as the free-rider. Because free-

riding nodes are not affected by this selection, we observe a large difference in the variation in delay between free-riding and non-free-riding nodes.

We observed significant variation in delay under topologies c and d. In these topologies, when one of the central nodes was selected as the free-rider, the non-free riders are partitioned into two sub-networks with either two or three nodes. Because free-riders do not propagate metadata, nodes partitioned by a free-rider do not detect metadata on the opposite side of the topology. This results in nodes only being able to demand content from adjacent nodes, which further results in abnormally low delay and high delivery ratios in some trials.

5.5 Field trial

We conduct a field trial to evaluate our prototype CTN system and the Laissez-faire framework under real-world conditions. Two high-level goals influenced the design of our field trial. Because we intend to release our system as open source software upon completion of this thesis, it is essential that serious software bugs be caught and fixed. The field trial served to validate our prototype implementation under unpredictable, real-world scenarios.

Our second goal was to evaluate the Laissez-faire framework subject to the mobility of real users. Since communication between nodes in Laissez-faire is dependent on the trust relationships between nodes, we chose to configure all nodes in the field trial to be mutually trusted. The resulting trust, and therefore, communication topology forms a clique graph. This enables all nodes in the field trial to communicate with each other, which maximizes the potential communication capacity of the network. For the purposes of experimentation, participants were not allowed to revoke trust for other nodes in the experiment. The trust topology was static throughout the field trial (with the exception of adding nodes as participants were given devices). Section 5.5.3 studies the population where nodes are not all mutually trusted.

5.5.1 Methodology

Our field trial consists of 36 participants recruited from the graduate student and faculty population of the School of Computer Science at the University of Waterloo. Each

participant was given a BlackBerry Bold 9000 smartphone, preloaded with the fully functioning prototype CTN BlackBerry application. We have included a slightly modified copy of the application tutorial distributed to our field trial participants in Appendix A. This document explains how the application works and describes each component of the application using screen captures. The field trial lasted for three weeks. During this period, participants were asked to keep the device with them at all times and to keep it charged.

Our prototype implementation allows users to select both their preferred node ranking strategy and the fragment selection strategy. However, we disabled the ability to change this setting during the field trial. Upon installation, each node uniformly randomly selects both a node ranking and fragment selection strategy. These choices are fixed for the remainder of the trial.

Upon installation, the application scans the device's SD card and indexes any content items that are present on the device. Our implementation of the Content Space subsequently monitors the BlackBerry file system for new content. Participants in the field trial were encouraged to take photos and record videos with the BlackBerry camera, record voice notes, or attach the device to their computer to load with content from the Internet. However, prior user trials have proved that users should not be depended upon to use an experimental application [144]. We therefore loaded the prototype with an additional component called the Field Trial Daemon. The daemon had four important roles. The first role was to create content for dissemination through the system. Every hour between the hours of 8am to 10pm, the daemon would generate a file of random data. The size of the file was uniformly randomly selected from the set of {1 MB, 2 MB, 4 MB, 8 MB, 16 MB}. Any content created by the user caused the 1-hour timer to reset. In a previous field trial of the MobiClique system, only a smart portion of users interacted with the application [143]. However, to our surprise, 19 of the 36 participants generated some content over the course of this field trial.

In our field trial, only 15 of the 36 participants manually expressed demand for a content item. Of the 15, nobody expressed demand for more than 13 items. The second role of the daemon is, therefore, to simulate user demand for content. Every hour the daemon is responsible for selecting a content item from the Content Space and setting its demand state to `true`. Should the user manually express demand for content, the hourly demand timer is reset. For verisimilitude, the selection process mimics the popularity distribution of content downloads observed in BitTorrent [203]. Each content item added to the Content

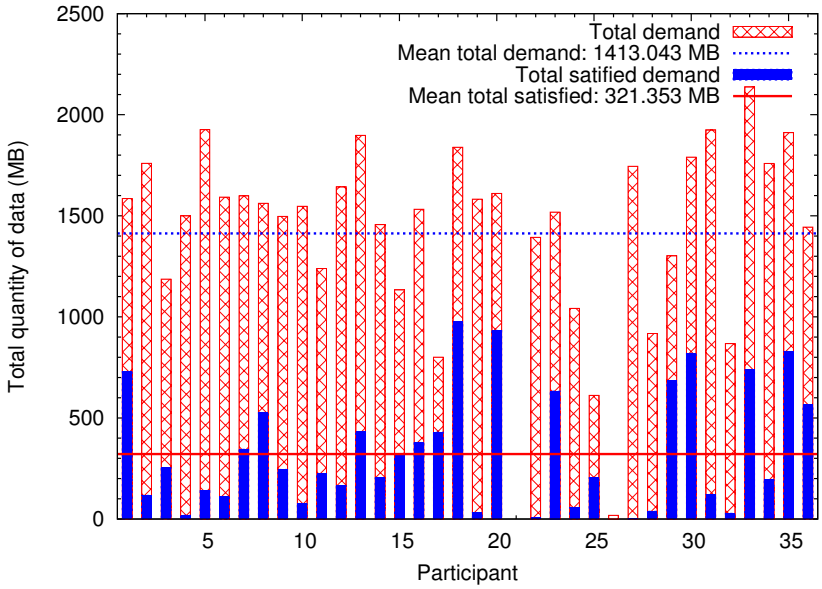
Space is assigned a uniformly random integer 'popularity' value between [0, 9] (0 being the highest popularity). The popularity value is embedded in the metadata of a content item that is stored on the device and is transferred to other nodes. When the daemon needs to demand a content item, it first selects a random integer using a Zipf distribution (exponent 0.66). The daemon then uniformly randomly selects a content item from the Content Space with the selected popularity. If no content items exist with the desired popularity, the process is repeated until an item is retrieved or there exists no items to demand. If there is no content to demand, this function of the daemon is suspended until a metadata update occurs and metadata corresponding to non-owned content is obtained.

The third role of the daemon is to upload the logs produced by the field trial to our experiment server. Logs are uploaded on an hourly basis, and allow us to monitor the health of every node. This let us capture software errors in the protocol stack within 1 hour of them occurring. Being able to rapidly identify and correct software errors was invaluable to ensuring the data derived from the field trial was valid. Once a software error was fixed, we uploaded a new application binary to the experiment HTTP server.

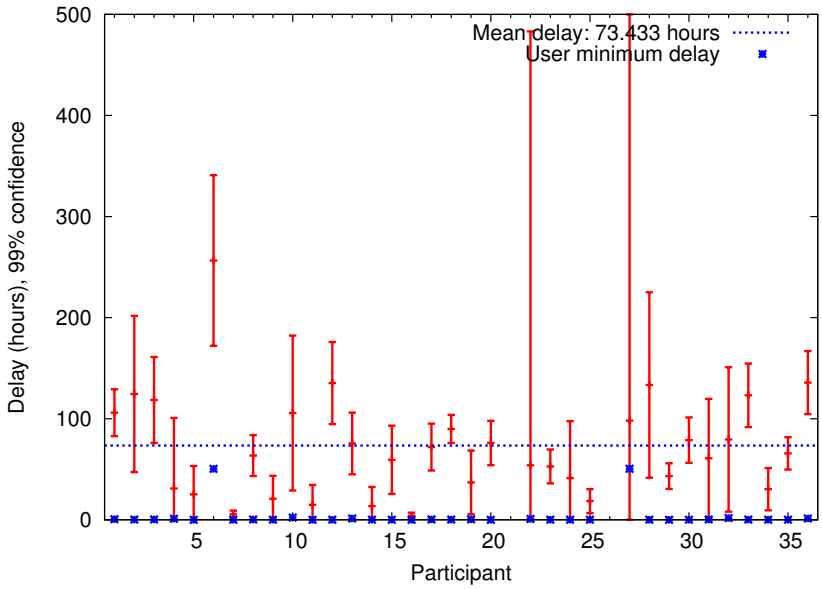
The final role of the daemon is to detect the presence of a software update by comparing the version of the application to the version number available for download. If a new version was detected, the daemon would notify the user of the update and to prompt them to download it. Upon download the application automatically installed and rebooted the device. This process would concurrently post the version number of the installed application so that we could monitor the deployment progress of an update. With few exceptions, most users observed the notification and downloaded the update within 3 hours of an update.

5.5.2 Analysis

Our field trial began Thursday, January 26, 2012 and terminated at noon on Friday, February 17, 2012. Over this period the 36 participating nodes demanded a total of 51.284 GB of data. Only 11.569 GB of data was successfully retrieved, resulting in an overall delivery ratio of 0.226. Although this may not sound like a large quantity of data, it is nearly an order of magnitude more data than what was exchanged by the average node per day during conference-based field trials of the MobiClique system [143]. The aggregate quantity of



(a) Quantity of content demanded and satisfied for each node.



(b) Mean delay to satisfy demand for each node.

Figure 5.8: Aggregate field trial participant statistics.

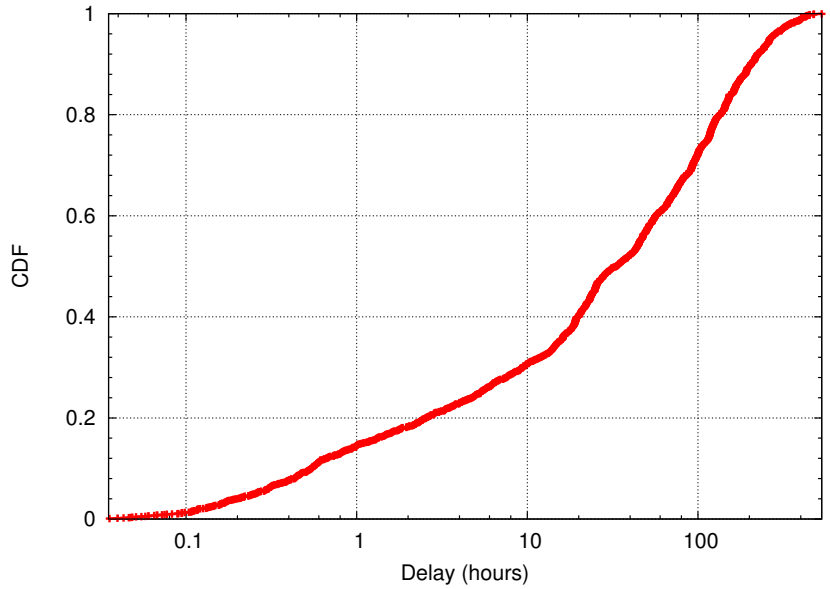
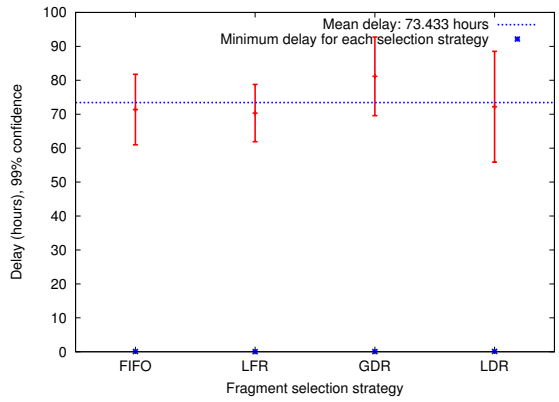


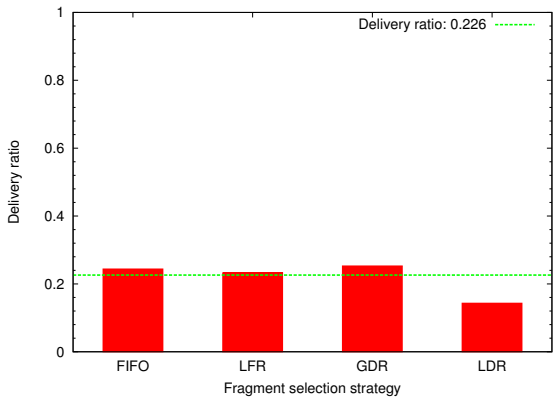
Figure 5.9: CDF of delay to satisfy demand over all content items.

content both demanded and satisfied for each node is illustrated in Figure 5.8(a). We observe that most nodes demanded approximately the same quantity of content since demand is automatically expressed on an hourly basis. Two nodes were isolated for entire duration of the experiment, and consequently did not obtain any metadata or demand any content. Interestingly, several highly connected nodes retrieved nearly 1 GB of content. We observe this cluster of nine nodes later in this section. Although this quantity of data is small in comparison to a typical BitTorrent download [203], we believe that it is reasonable for an initial deployment of an experimental system. We have included the traces in Appendix B that illustrate the timeline of when demand was expressed and when demand is satisfied. When demand is expressed, the quantity of outstanding demand is increased by the size of the content item. When demand is satisfied the outstanding demand decreases. We encourage to interested reader to review the timelines to better appreciate the limitations of these results.

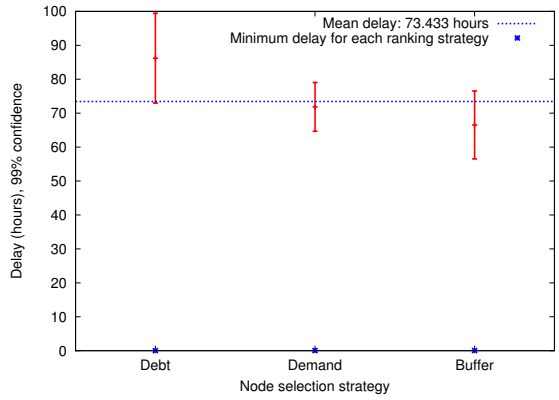
We observed a global mean delay of 73.433 hours. The delay for each user is illustrated in Figure 5.8(b). The large error bars correspond to nodes that received very few content items. Although the global average is quite high, it is important to note that most nodes



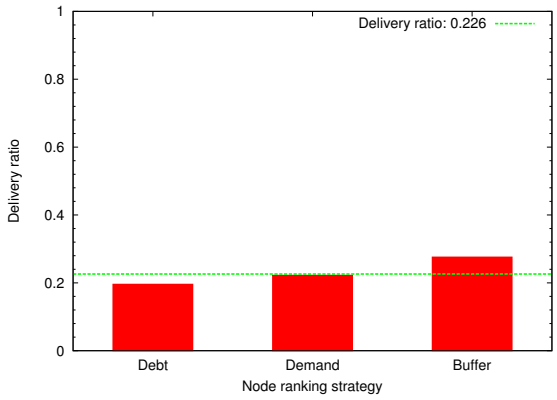
(a) Mean delay to satisfy demand by fragment selection strategy.



(b) Delivery ratio by fragment selection strategy.



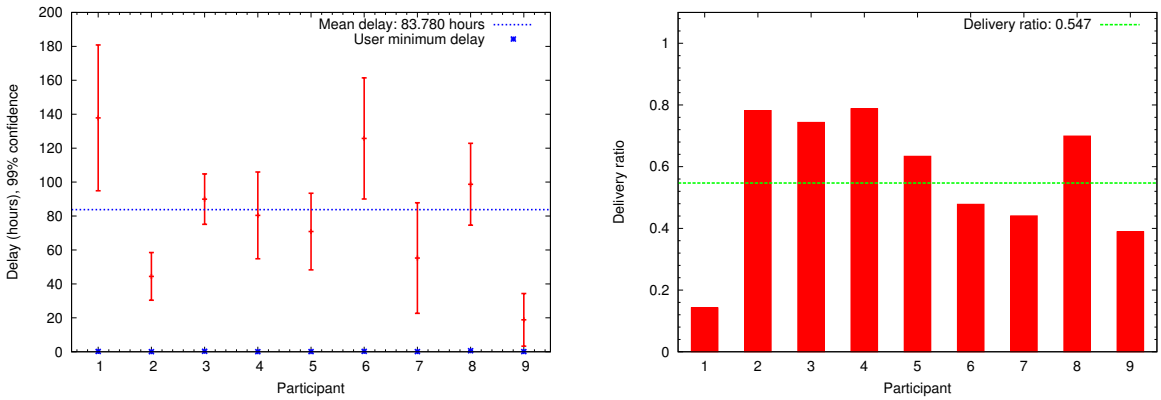
(c) Mean delay to satisfy demand by node ranking strategy.



(d) Delivery ratio by node ranking strategy.

Figure 5.10: Analysis of delay and delivery ratio vs. node ranking and fragment selection strategies.

received some content items within a few hour. This result is apparent in the CDF of delay over all delivered content items that we illustrated in Figure 5.9.



(a) Mean delay to satisfy demand for each node.

(b) Delivery ratio for each node.

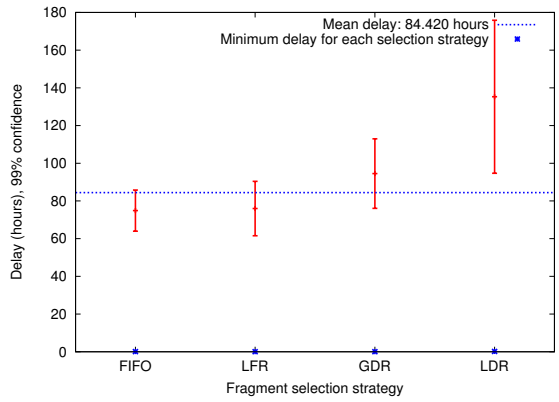
Figure 5.11: Analysis of delay and delivery ratio for each node in the nine-node cluster.

Impact of node ranking and fragment selection strategies

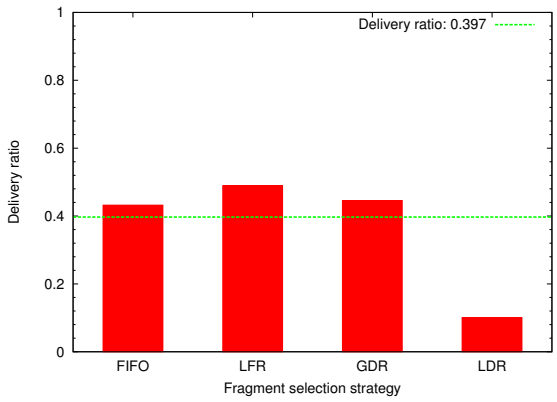
Overall, we observed that least-data-remaining fragment selection strategy resulted in a delivery ratio of approximately half the other fragment selection strategy. However, we observed no significant difference in delay between the four fragment selection strategies. Figures 5.10(a) and Figure 5.10(b) illustrates these two results. When selecting content to retrieve on behalf of others, ranking nodes using the buffer strategy both increases the delivery ratio and reduces delay! No fragments were discarded from any device over the three-week period. We are therefore unable to assess the effect of node ranking or fragment selection strategies on drop ratio at this time.

Highly connected clusters

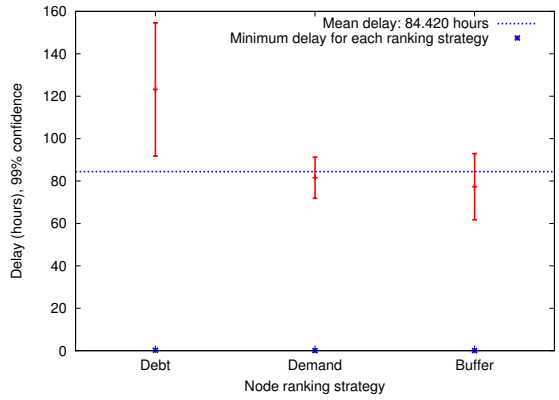
We briefly examine the performance of the system for two clusters of highly connected users. We believe that these clusters of nodes represent each of the CTN use cases outlined in Section 1.2.1. For the purposes of this discussion, we refer to nodes within the cluster as 'internal' nodes and nodes outside of the cluster as 'external' nodes. The 'internal delay' is the delay in satisfying a demand for content that resides on a node within the cluster. Similarly the 'internal delivery ratio' is the delivery ratio of content demanded from nodes within the cluster.



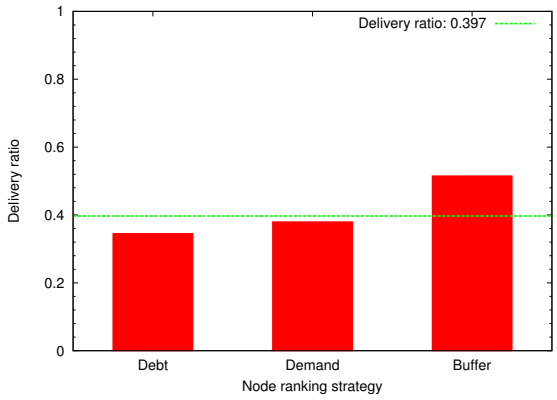
(a) Mean delay to satisfy demand by fragment selection strategy.



(b) Delivery ratio by fragment selection strategy.



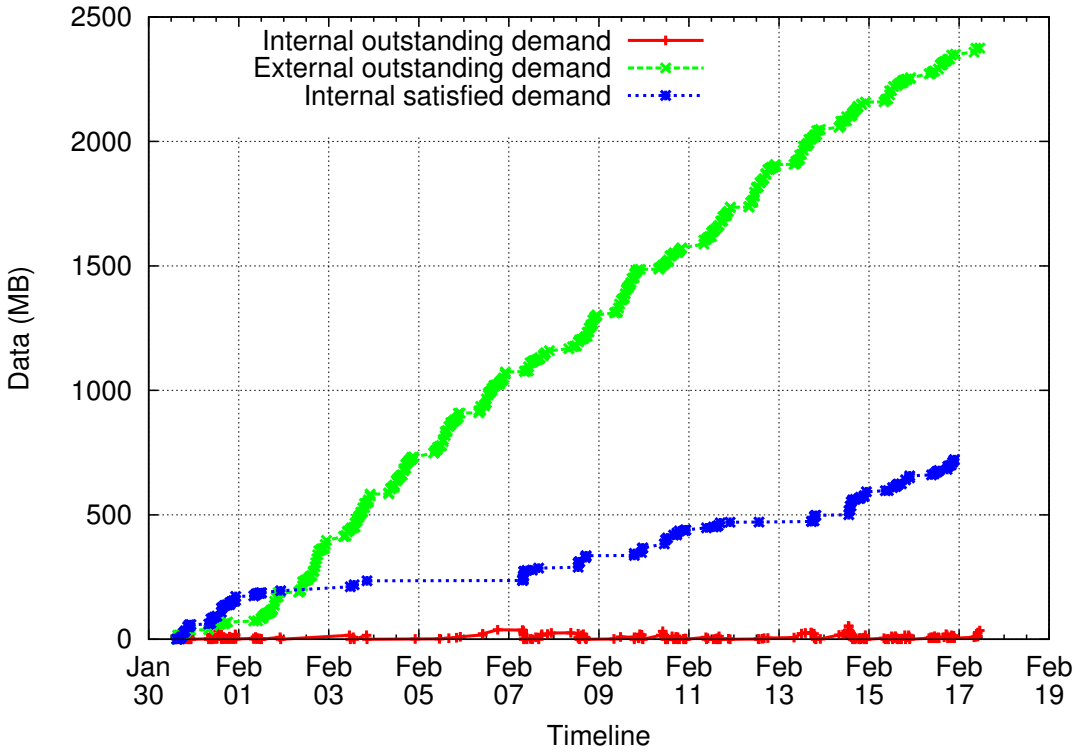
(c) Mean delay to satisfy demand by node ranking strategy.



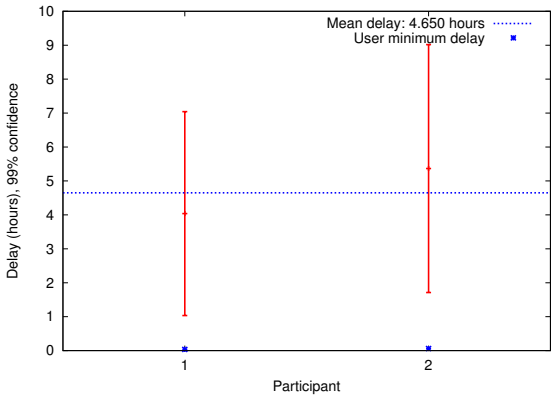
(d) Delivery ratio by node ranking strategy.

Figure 5.12: Analysis of nine-node cluster.

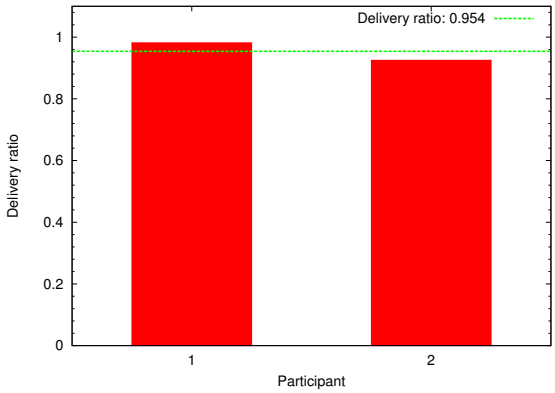
The first cluster corresponds to a research lab with nine students that were actively engaged in the study. These participants were the source of most of the user-generated content. Most user-initiated demands for content came from this group. One student in this cluster spent a significant amount of time at home. Another arrived very late in the day, which reduced the collocation duration with other participants. Overall this cluster demanded 8.787 GB of content and retrieved 4.806 GB to achieve a delivery ratio of 0.547 - over twice the global average! The internal delivery ratio for each of the nine nodes is



(a) Internal and external outstanding and satisfied demand timeline.



(b) Mean delay to satisfy demand.



(c) Delivery ratio of content demanded from within the two-node cluster.

Figure 5.13: Analysis of an isolated two-node cluster.

illustrated in Figure 5.11(b). Interestingly, the increase in delivery ratio is accompanied by a corresponding increase in internal delay, as illustrated in Figure 5.11(a).

Figure 5.12 illustrates the delivery ratio and delay for each fragment selection and node ranking strategy for this close-knit cluster. Again we observe that the least-data-remaining fragment selection strategy results in an extremely poor delivery ratio. This is due to its preference for small files, which starves the retrieval progress of larger files. We therefore conclude that when a node is retrieving fragments for itself, it should utilize either the FIFO or least-fragment-remaining fragment selection strategy. When retrieving content on behalf of others, a node should utilize the buffer strategy to rank nodes. A strategy that is biased towards nodes that are expected to be encountered the soonest results in more content being delivered with lower delay.

The second cluster corresponds to two collocated study participants that did not detect or exchange content with any other nodes except for the node operated by the study organizer. These two participants were the last to join the experiment. When distributing the BlackBerry devices to these two participants we seeded one of the nodes with nearly 3 GBs of known metadata. Unfortunately, their isolation prevented any external content demands from being satisfied. This trend is illustrated in Figure 5.13(a).

Unlike the previous clusters, delay is this cluster is very low - driven up by long period of disconnection between February 4th to 7th as observed in Figure 5.13(a). Mean delay was 4.65 hours. The minimum delay among these nodes was an impressive 126.47 and 216.54 seconds. These delays obviously correspond to content that was demanded while the other node was collocated. This result is illustrated in Figure 5.13(b). Not surprisingly, the internal delivery ratios for these two well-connected nodes is 0.95 (Figure 5.13(c)). The reason why they are not 1.0 is because the nodes were separated prior to their collection at the end of the experiment. If they had detected each other, they would have immediately satisfied each other's internal demands.

The role of an SMS-based control channel

In the previous chapter we introduced the SMS Connection object as an optional control channel in our prototype CTN system. The use of SMS, which uses cellular network infrastructure, clearly violates our design goals by providing a means for attackers to

identify CTN participants. The security repercussions of using SMS will be revisited in Chapter 8.

Although our current prototype implementation does not depend on SMS, we believe that SMS could have great value to users that are not engaged in nefarious activities and are, presumably, less concerned with the protection of their identity[2]. Nodes could, for example, use SMS to signal changes to their Content Space or disseminate updated demand vectors when fragments are retrieved or demands change. Future applications of an SMS-based control channel are further discussed as future work in Chapter 8.

This section provides empirical data to quantify the value of an SMS-based control channel in the system. We observe that a total of 691 control messages were exchanged during the field trial. The vast majority of these messages originated from the study organizer for experimentation purposes when adding new nodes to the system. These control messages would not be used in a real deployment. However, they have same characteristics as control messages used in a non-experimental setting. In total, 595 control messages originated from the study organizer to initialize and configure the experiment. The remainder of the control messages correspond to several active participants sending text messages to each other. We observed a median control message delivery delay of 87.32 seconds. This value was primarily influenced by a one-minute delivery deadline set by the Client Core. When a new control message is created, it is first scheduled to be delivered over an opportunistic connection. If after one minute the recipient node has not be encountered then the control message is scheduled for transmission over the SMS-based control channel. The polling interval of the Internet-based SMS Handler on the receiving node, that was set to 30 seconds, also introduces delay. The mean value was significantly higher due to control messages that were created while the recipient nodes were disconnected from the Internet and thus unable to fetch messages from our Internet-based SMSC. The delivery ratio of control messages was 1.0.

We assume that nodes cannot be relied upon to deliver messages for others. Therefore, control messages are better delivered directly from the source to destination node. We now evaluate the delay and delivery ratios of the 691 control messages in an emulated scenario where an end-to-end SMS-based control channel does not exist. In this experiment, we extract the time that each pair of nodes contacts each other during the field trial. We

[2]In practice, the use of CTNs for non-nefarious activity should be encouraged because it would provide some degree of plausible deniability for users that are successfully identified as participating in a CTN.

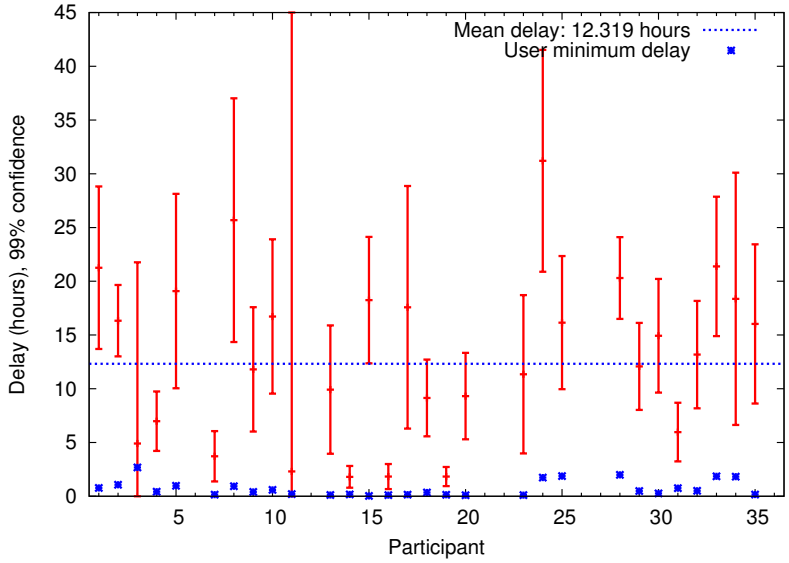
also extract from the field trial traces the creation time of each control message. Using the creation time of a control message and the contact traces between nodes, we emulate the process of delivering control messages in the absence of end-to-end connectivity. This emulation assumes that all control messages between a source and destination node are delivered during an opportunistic connection.

For control messages sent from the study organizer, we observed a delivery ratio of only 0.32. This low value is influenced by several nodes that were never detected by the study organizer after being initially added to the experiment. A control message that was custom to the experiment was created for each subsequent new member. Not contacting nodes that were added to the experiment at an earlier time caused the control messages to never be delivered. Of the messages that would have been delivered, we observed a mean delivery delay of 3.92 days[3]! The text messages exchanged between nodes had a mean delay of 7.24 hours and a delivery ratio of 1.0. This is due to the physical proximity of the participants that sent the messages. Messages that were sent during the day would be delivered quickly. Messages sent at night would be delayed until the following day.

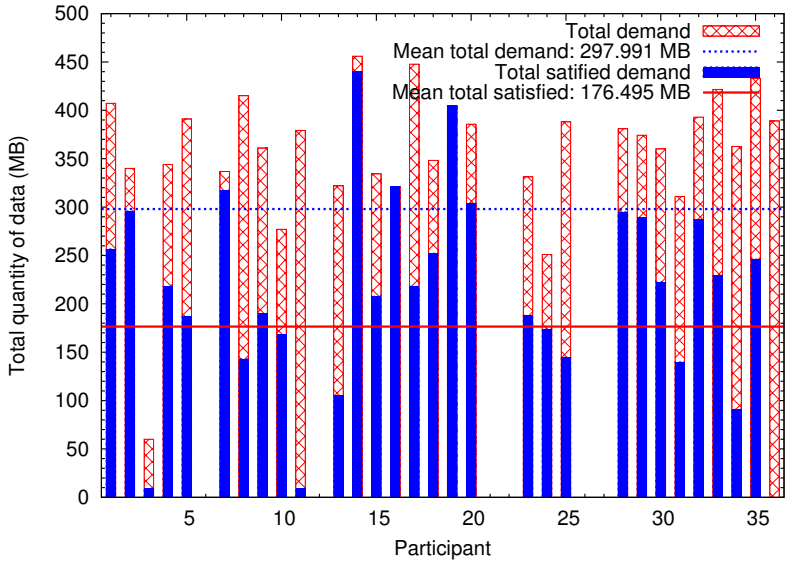
5.5.3 Analysis of non-global trust

In both our controlled experiments and the field trial, all nodes were configured to be mutually trusted, and thus all nodes were capable of communicating with each other. Following our field trial, we conduct a final one-week controlled experiment that segregated nodes into eight groups, where the nodes in each group are mutually trusted. Our prior work segregated users based on their interests on Facebook [144]. Unfortunately, the demographics of our field trial participants did not facilitate this comparison. Instead, we segregate nodes based on their research area. In the absence of any prior work (to the best of our knowledge) that examines how users in a similar network, under the same conditions, would organize themselves, we believe our approach is a reasonable compromise. We then connect trust groups based on known social relationships between participants. In the absence of social ties, nodes with the lowest inter-contact time were designated mutually trusted nodes.

[3]This value was artificially inflated by the study organizer periodically walking around the building at night to verify that participants took their devices home.



(a) Mean delay to satisfy demand for each node.



(b) Delivery ratio for each node.

Figure 5.14: Analysis of delay and delivery ratio for each node segregated into different trust groups.

Recall that our experiment framework allows us to specify discrete contact events. We use this feature to 'playback' the contact traces from our field trial. Unlike our previous controlled experiments in Section 5.4 that used a repeating contact interval, contact events in this experiment correspond to actual contact events that occurred during the field trial. However, given that many nodes are now placed in disjoint trust groups, many contact events may have no consequence because the nodes may be non-trusted and thus unknown.

Overall, we observed a significant reduction in mean delay! The delay for each node is illustrated in Figure 5.14(a). The mean delivery ratio of 0.592 is also significantly higher than the delivery ratio of 0.226 observed in our previous field trial. However, the reduced delay and higher delivery ratio comes at a cost of significantly less data exchanged between nodes. The quantity of data exchanged for each node is illustrated in Figure 5.14(b). Due to the constrained trust topology, connection opportunities that existed in the previous trial were not possible in this experiment. The scope of metadata dissemination is consequently lower, forcing nodes to demand content that is present on the devices of fellow group members. Since trust was artificially derived from nodes of participants in the same research area, most group members are physically collocated. The combination of these properties is responsible for these gains.

5.6 Chapter summary

This chapter has introduced the concept of a hands-off approach to DTN routing. Our approach differs from existing DTN routing protocols that mandate specific node behaviour that is generally intended to maximize the aggregate performance of the system, rather than the performance experienced by a specific node. Participation in these networks requires a mobile device to allocate storage, energy, and other limited resources, for the benefit of others. These resources are often highly constrained [129], and will continue to be for the foreseeable future. We believe that the network-centric approach to resource allocation used in existing work is incompatible with users' expectations of how resources on their personal device should be consumed. Further, if users are able to modify the source code of the network, a major design goal of our work, they do so for their own selfish benefit.

Under Laissez-faire, nodes operate independently, and presumably selfishly, to maxi-

mize their own utility. We believe that a selfish, individualist approach to routing, where each the node can control how their own resources are allocated is fundamental to the practical deployment of open source DTN systems targeted to average smartphone users that are sensitive to resource costs.

We present a set of basic operations that allow nodes to exchange control messages, request fragments, and propagate both content metadata and nodes' demands for it. Participation in our prototype CTN system only requires that nodes implement this simple tit-for-tat protocol.

We proposed several example strategies for ranking nodes and selecting fragments. Although nodes are free to implement their own strategies in the pursuit of maximizing their own utility, we found that the FIFO and least-fragment-remaining selection strategies performed best. We also found that when retrieving content on behalf of other nodes, the buffer node ranking strategy that favours nodes that are frequently contacted performs best.

Finally, we showed how an optional SMS-based control channel can be used to propagate small amounts of control information between nodes. Although, our current prototype does not depend on the control channel, we outline several areas of future work where we believe that a control channel would offer significant benefit in Chapter 8. In the next chapter we explore the design of a transport protocol that underlines the SMS-based control channel.

Chapter 6

Design of an SMS-based Control Channel

In the previous chapter we demonstrated that an SMS-based control channel can both reduce delay and increase the delivery ratio of control messages within a CTN. This chapter motivates the design of a transport protocol underlying an SMS-based control channel through a characterization of SMS on a major Canadian cellular network. We then evaluate the performance of the transport protocol under a range of network properties.

6.1 Introduction

The Short Message Service (SMS) is a phenomenally popular global wireless service. In 2010 alone, nearly 6.9 trillion SMS messages were sent [145]. This number is expected to climb to 8.7 trillion by 2015 [78]. 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 notifications, search the Internet, and exchange status updates with servers on the Internet [50]. This list of uses promises to grow with no sign of slowing down.

We build an efficient and reliable data transport protocol on top of SMS for use as a control channel within a CTN. SMS provides end-to-end connectivity among otherwise disconnected CTN nodes that choose to enable it. This chapter makes two major contributions:

1. We motivate the design of our transport protocol through a characterization of the SMS service of a major Canadian cellular network provider. We explore 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, and message reordering is common. 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 work differentiates itself from prior work that has focused purely on aggregate statistics observed

service providers. Our measurement tools are freely available for further study of other cellular networks [124].

2. 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. 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 is licenced under the Apache Licence and can be downloaded from our website [124]. 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.

The remainder of this chapter presents an overview of SMS and prior work. Section 6.3 describes our measurement study and we present out experimental results in Section 6.4. Section 6.5 describes the design of SMS-TP. We evaluate the protocol in Section 6.6. We summarize the chapter in Section 6.7.

6.2 Motivation

6.2.1 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 [63]. Moreover, alternatives to SMS, such as the Enhanced Message Service and GPRS/EDGE, are poorly supported or sparsely deployed outside of developed regions [94].

- **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 [81, 99]. 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 [125]. SMS *Gateways*, which bridge the Internet and cellular networks, have also proliferated, thus enabling applications on the Internet to easily send SMS messages to 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 [31, 114, 177, 184]. Applications that must exchange more than 140 bytes of data do so by fragmenting their data and sending it using a series of messages [168, 179]. These applications use a simple stop-and-wait protocol, which we describe in detail in Section 6.5.1. As mobile devices continue to become an integral part of daily life [92], 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 message stream, modeled as a communication channel.

6.2.2 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 Enhanced Message Service (EMS). As a data channel, SMS is greatly inferior to cellular data services such as EVDO, and GPRS/EDGE. SMS has significantly lower data rates, higher 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 [94], leaving SMS as the only cost-effective, viable option for data communication.

EMS is an application level extension to SMS [1]. 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 SMS 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.

6.2.3 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 References [48, 183, 196]. Other work examines the increased load on the cellular network and the GSM signaling channel introduced by SMS [57, 71, 72, 120].

Zerfos et al. examine the characteristics of SMS from the perspective of a cellular service provider [115, 201]. In this work, Zerfos et al. present 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 et al. classify the 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 timer 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, which are 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 et al. study by examining SMS under bursty workloads and by examining the root causes of delay variation.

6.3 Characterization

The goal of this section is to define the channel characteristics of SMS, where this channel models a stream of SMS messages sent from a source to a destination. 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.

6.3.1 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 in delay inherent to the cellular network and to motivate our experimental approach. Although this discussion is made in the context of GSM, SMS operates in a similar fashion over CDMA-based networks [157].

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) with which it is currently associated. Since the random access channel is a shared channel using slotted ALOHA, communicating over it can introduce random back-off delays. The *base station controller* responds to the request by allocating a dedicated control 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 [49, 136, 160].

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 [160].

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 that 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 control channel over the random access channel as before. Once the MS_{dest} receives a dedicated control 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}.

6.3.2 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.

Transmission time

The transmission of an SMS message requires that the MS first allocate a dedicated control 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 between the MS and base station. If the signal strength is too low, communication between the MS and base station may be intermittent or fail entirely. Transmission time is also affected by delays introduced by random back-off when requesting a channel over the random access channel [49, 136]. 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 control channel, and to successfully transmit the message. This process is illustrated in Figure 6.1. Occasionally a message may fail to transmit due to

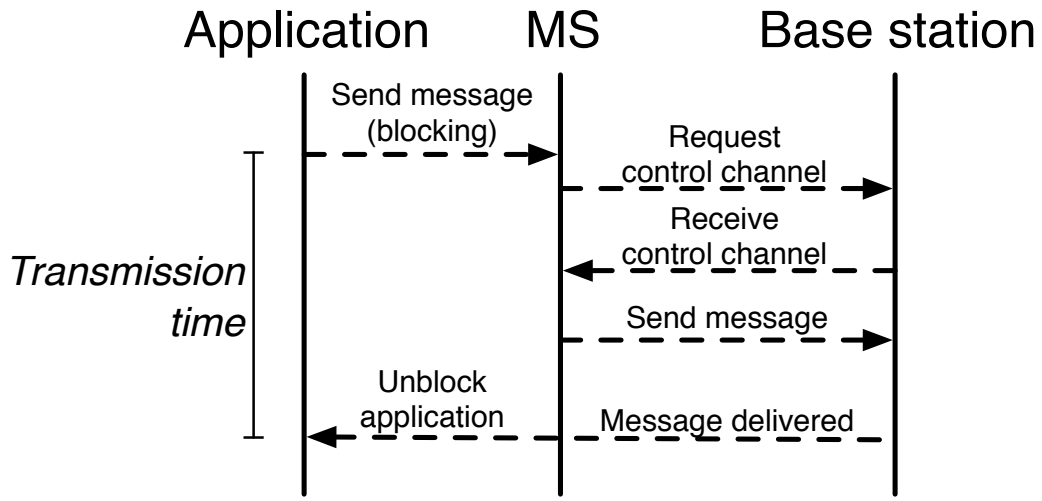


Figure 6.1: Evaluation of the transmission time of an SMS message.

channel contention on the random access channel or failure to allocate a dedicated control channel. Our study also examines the transmission failure rate and the time required for a transmission failure to be detected.

Delay

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 control 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.

Loss rate

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

control 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}.

Message reordering

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 [135], where 'late' packets are declared reordered. Our second reordering metric considers the overall delay introduced as a result of reordering. We refer to this as *reordering delay*. For monotonically increasing message IDs, reordering delay is calculated as the difference in time between the arrival of a reordered message i and the minimum receive time of message $i+1, i+2, ...$ 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 buffered while 3 was delivered.

6.3.3 Methodology

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

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 experimental setup for the tethered cell phone case is illustrated in Figure 6.2. 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

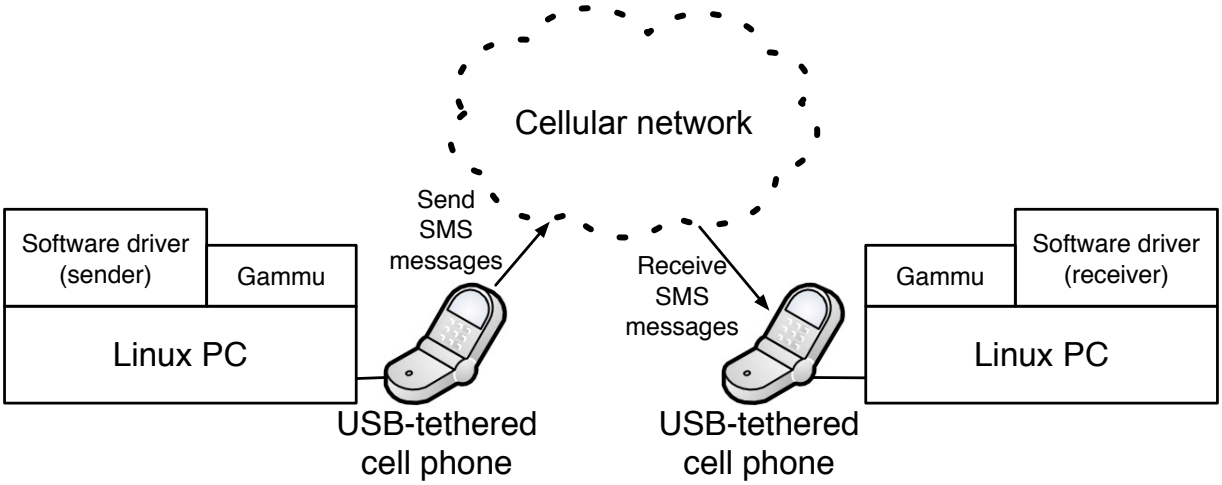


Figure 6.2: Tethered cell phone experimental setup.

BlackBerry also contains a different cellular radio, which allows us to assess the effect that the device may have on SMS service. The Pearl contains a MMM6000 radio from FreeScale Semiconductor with a Sky77526 power amplifier from Skyworks. The 8820 also contains a MMM6000 radio and a Freescale PMM6037 power amplifier. The Bold contains an RPF 09040B radio from Marvell Technology that supports 3G [6]. We equip each device with a SIM card from a major Canadian GSM service provider (Rogers Wireless [7]). 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 even in developed regions will exhibit different properties [201]. We will revisit the properties of alternate networks through simulation later in this chapter.

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 collects

statistics and manages each SIM card's SMS quota[1]. 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 [4].

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 2 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 affect 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.

Parameters

The performance characteristics of a stream of SMS messages 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 control channel it if 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 introduce delays depending 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 6.3.

[1]Unlimited SMS packages did not exist in Canada in 2008 when these experiments were performed.

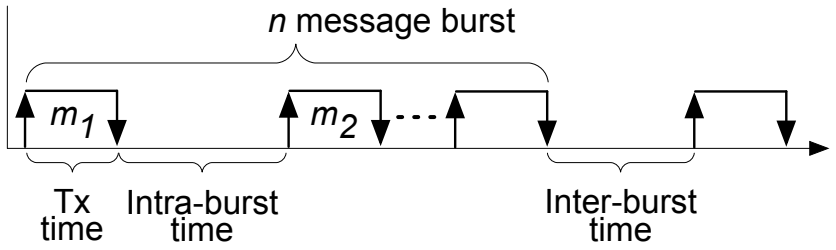


Figure 6.3: Relationship between intra-burst and inter-burst times.

- **Device**: Different cellular radio technologies have different tolerance for low signal strength and interference. 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 control 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 may impact SMS performance. The process of changing base stations may also affect SMS.
- **Local subscriber traffic**: Without access to service provider's records[2] it is impossible for us to measure the effect that subscribers have on each other. We assume that

[2]Although some service providers have been willing to share data [201], 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 is [93].

Name	Duration	Number of messages	Intra-burst time(s)	Inter-burst time
Network Variation	4 weeks	4 devices x 4 weeks x 55 messages per hour = 36,960	500 ms	1 hour - (time to transmit all 55 messages)
Message Timing	4 weeks	4 intra-burst times x 1 week x 55 messages per hour = 36,960	1 sec, 2 sec, 10 sec, 20 sec	1 hour - (time to transmit all 55 messages)
Message Index	Not applicable	4 x 1000 messages = 4,000	500 ms	Not applicable

Table 6.1: Summary of experiments.

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 control channels. We compensate for local fluctuations by experimenting over long periods.

- **Cross traffic**: Cross traffic within the cellular network may cause excessive loads at the service center. In this study, we 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 [81]. 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.

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 interference. The devices are 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. These experiments are summarized in Table 6.1.

Network Variation Experiment: The first experiment evaluates the effect of the time-of-day and the device used. This experiment consists of four week-long trials. In each trial we use a different device to transmit bursts of 55 messages per hour over the entire week. The value of 55 messages per hour creates 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 is configured to be $3600 - \sum_{i=0}^{55}(t_i + 0.5)$ seconds to ensure that 55 messages are transmitted each hour.

Message Timing Experiment: Our second experiment examines the effect of intra-burst time by performing four additional week-long trials with varying intra-burst times. In this experiment we evaluate four additional intra-burst times: one, two, five, 10, and 20 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 control for the effect of time-of-day. Similarly, in this experiment we only use the 8820 devices to control for the device.

Message Index Experiment: Our final experiment examines the effect of the transmission index. The previous two experiments are 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 second 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 wrong [9, 115], it is not feasible to conduct experiments every day of the year. This assumption is therefore necessary to assess the other input variables. To avoid 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 [1], SMS messages can be transmitted on one of 65536 ports.

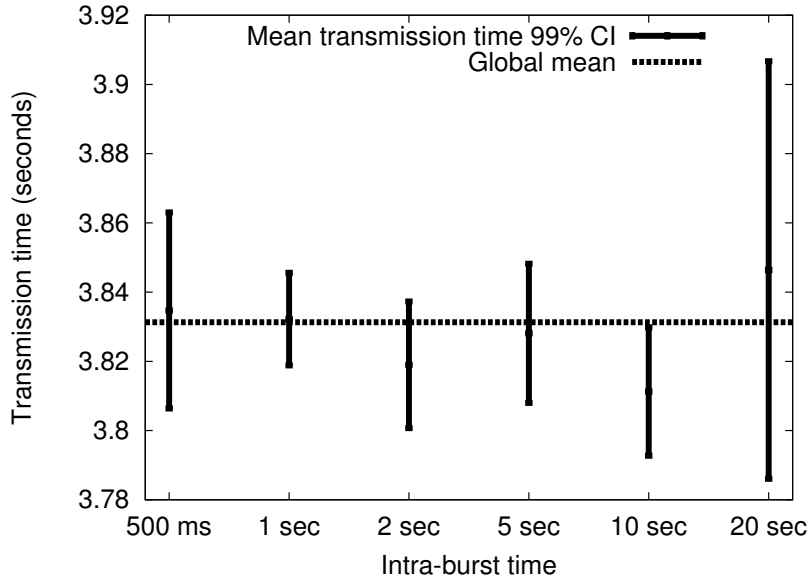


Figure 6.4: Mean transmission time with respect to intra-burst time. Note the magnified y-axis.

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 demonstrated 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.

6.4 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.

6.4.1 Transmission time

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

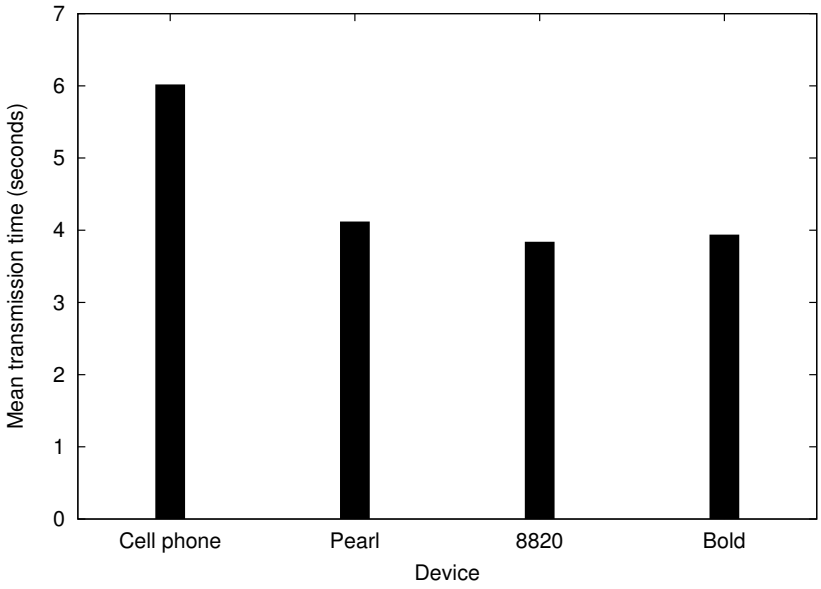


Figure 6.5: Mean transmission time for each device.

Intra-burst time: We evaluate the effect of intra-burst time on transmission time by performing a series of student 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 6.4. 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. Recall that the week-long trial consisted of sending 55 messages per hour over an entire week, and the second experiment consists of four additional trials where 1000 messages are exchanged between a pair of BlackBerry 8820 devices. 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.

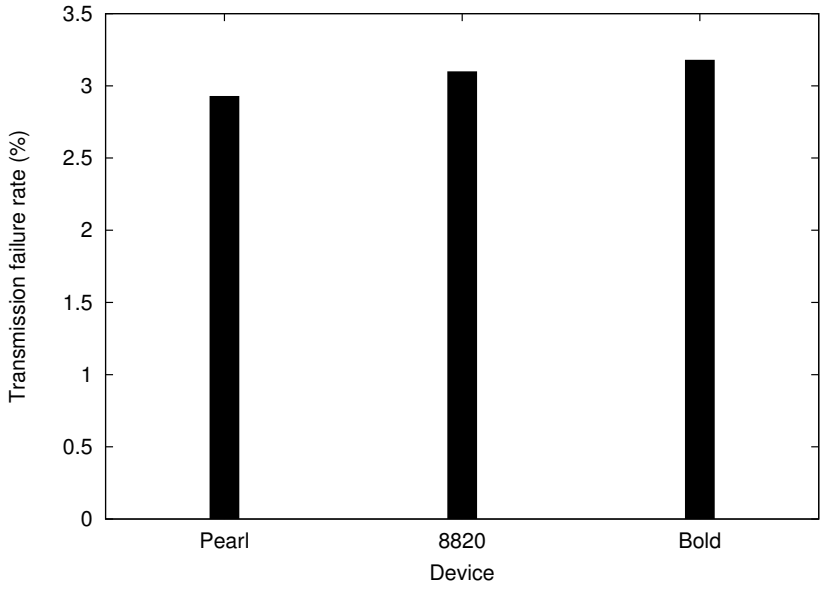


Figure 6.6: Transmission failure rate for each device.

For transmission time to be independent of the transmission index, we expect to see more variability within samples 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.

Device: 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. These results are illustrated in Figure 6.5. Using a student 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

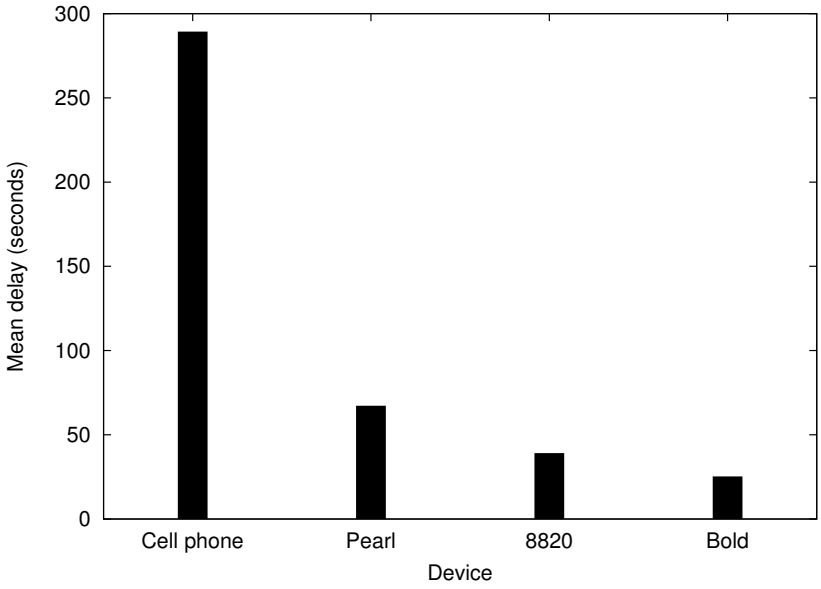


Figure 6.7: Mean delay for each device.

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 identical. These results are illustrated in Figure 6.6. 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 control 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 only 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 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.

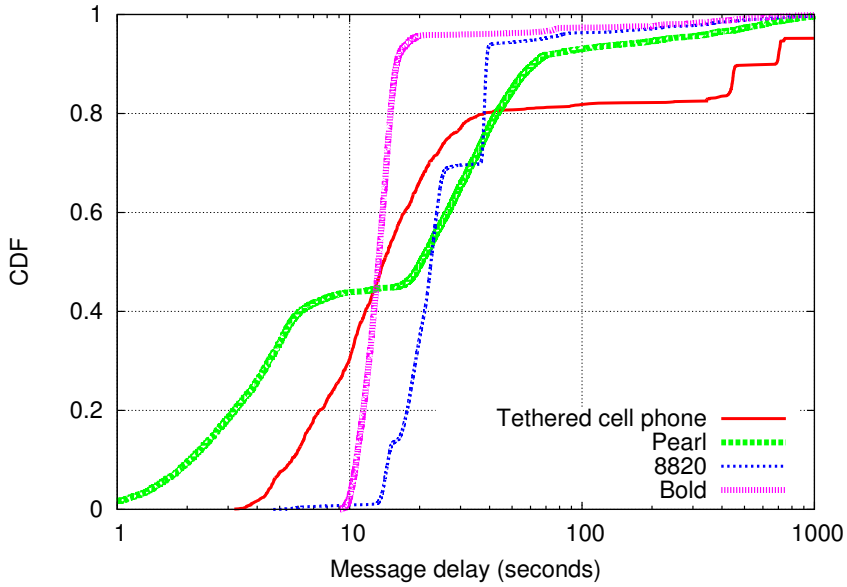


Figure 6.8: CDF for SMS message delay.

6.4.2 Delay

Because SMS messages are delivered sequentially, delay is tightly coupled with reordering. That is, messages may be delayed as a consequence of being reordered in a delivery queue. The relationship between delay and reordering is examined in Section 6.4.4.

Overall, in 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. These results are illustrated in Figure 6.7. With over 9,000 samples each, this indicates a significant correlation between delay and the device used. These results are illustrated as a delay CDF in Figure 6.8.

We also found that delay was correlated with the intra-burst time and the time-of-day, as discussed next.

Intra-burst time: Using the 8820, we found that the mean delay for messages with a 500 ms intra-burst time was 48.80 seconds. Mean delay falls to 39.12 seconds using a one-second intra-burst time and stabilizes at 37.57 seconds using an intra-burst time of two seconds; however, the mean delay of the two-second case is still within the 99% CI of

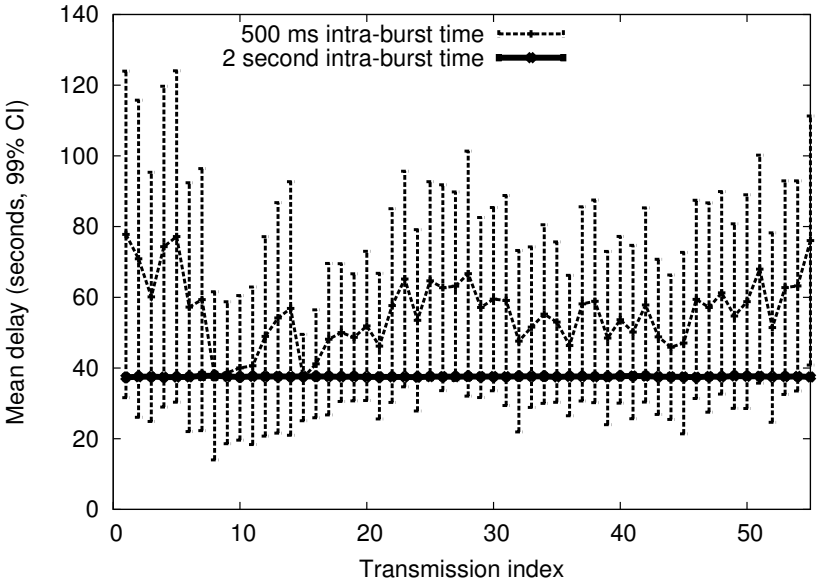


Figure 6.9: Mean message delay for 500 ms and two-second intra-burst times.

the 500ms case. These results are illustrated in Figure 6.9. Interestingly, we also observe, though this result is not shown, that intra-burst times higher than two seconds offer no improvement in delay.

A service center is capable of delivering multiple messages over one dedicated control channel [49], we examine how delay affects the rate that messages arrive at the receiver. Surprisingly, despite high delays within the network, bursts of messages are almost always delivered at a rate of one message every 4 to 6 seconds, independent of the intra-burst time. We illustrate this finding through the CDF of message inter-arrival time in Figure 6.10. We observe that messages sent with an intra-burst time of 500 ms may be classified into three groups based on their inter-arrival time at the receiver. Approximately 80% of messages arrive within 5 seconds of another messages. A second group consisting of approximately 18% of messages arrive between 5 and 15 seconds after another message. We therefore note that, in the worst case using a 500 ms intra-burst time, 98% of messages arrive at the receiver within approximately three times the median inter-arrival time. This message inter-arrival behaviour has the effective appearance of messages arriving in *batches*, with groups of messages with low inter-arrival times separated by messages with larger

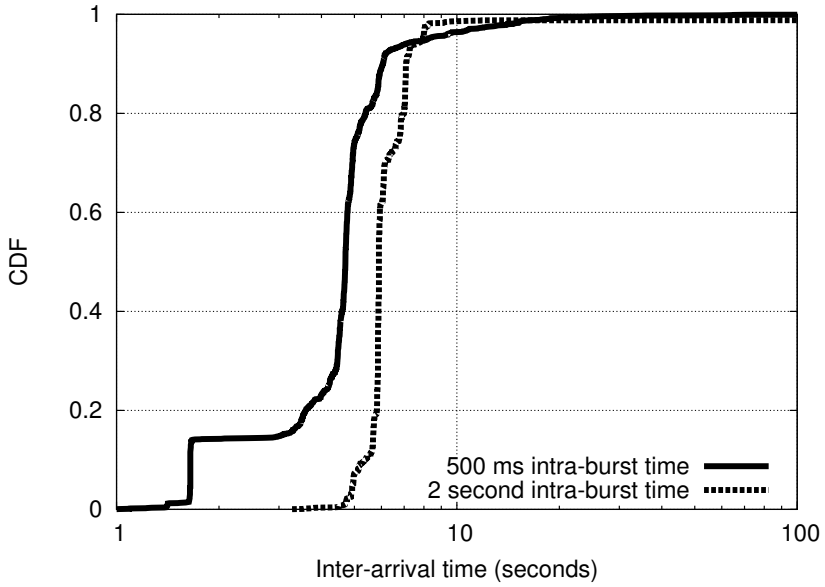


Figure 6.10: CDF of inter-arrival time (8820 using 500ms and 2 second intra-burst times).

inter-arrival times. We exploit this observation in the design of our transport protocol in Section 6.5.2.

The remaining 2% of messages have an inter-arrival time greater than 15 seconds. The high inter-arrival time of these messages is due to reordered messages that suffer abnormally high delays. We examine reordering in Section 6.4.4. Although we are unable to determine when a device establishes and releases a dedicated control channel, we believe that the consistency in inter-arrival times illustrated in Figure 6.10 is strong evidence that the device takes advantage of the channel to retrieve as many available messages as possible.

Time-of-day: The mean delay varies significantly over the course of a day. Across all trials, we observed that delay is lowest during the hours of 8 pm to 9 am. On the 8820, mean delay over 11 hour 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 variation in mean delay over the day in Figure 6.11.

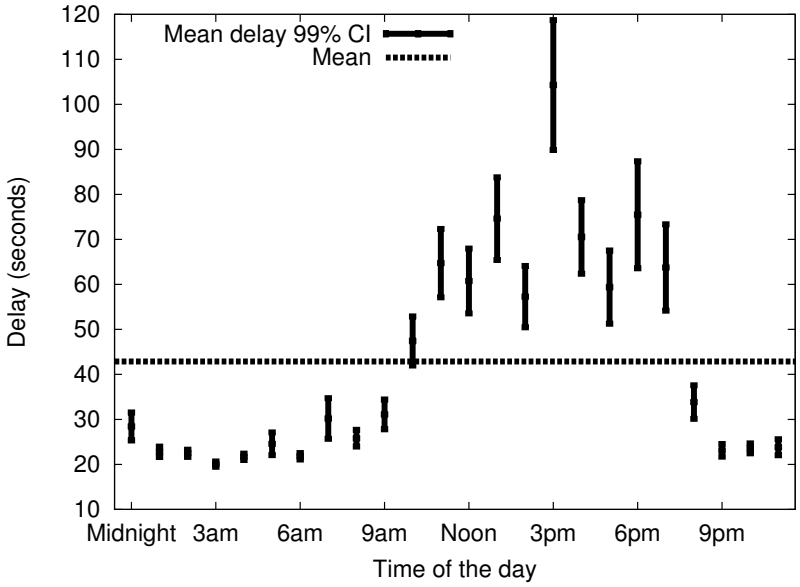


Figure 6.11: Mean message delay over the day (8820 using a 500ms intra-burst time).

6.4.3 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 essentially 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 Reference [115]. 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 believe that the device was unable to respond to paging requests by the cellular network. Then, after a series of failed retransmissions, the messages that expired within the service center and were deleted. Communication errors with the cell phone also caused 2.91% of messages to be duplicated.

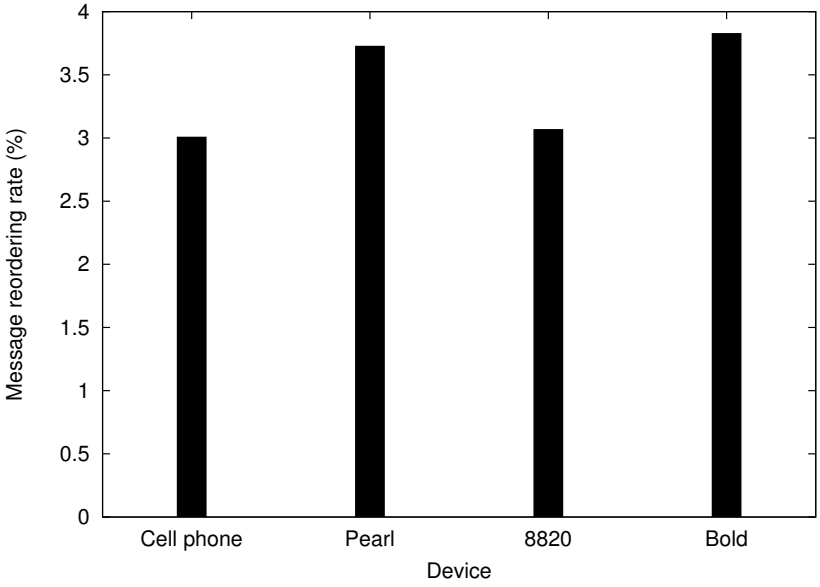


Figure 6.12: Message reordering rate for each device.

6.4.4 Reordering

Throughout all trials using an intra-burst time of 500 ms, we observed an overall 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. These results are illustrated in Figure 6.12. Given the small relative between the four devices, we assume that message reordering is independent of the device. Our analysis of reordering, therefore, aggregates 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 at either the sender or receiver. For clarity, this classification technique is illustrated in Figure 6.13. 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 in a single cell was 33.71 seconds.

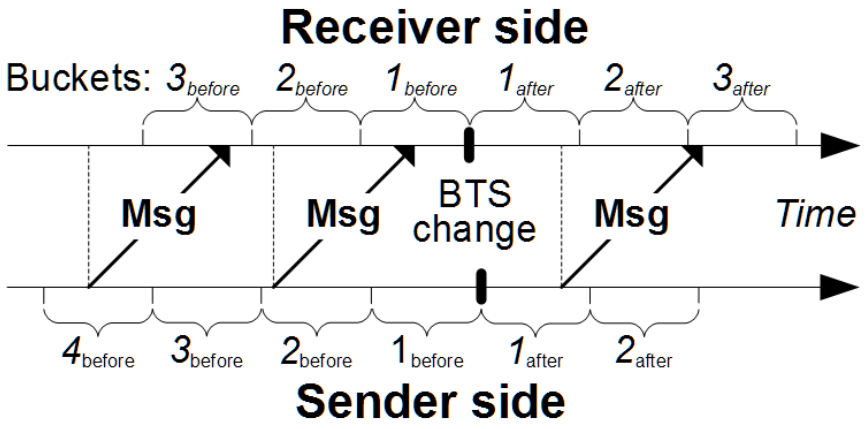


Figure 6.13: Method for classifying messages with respect to the time of a base station change.

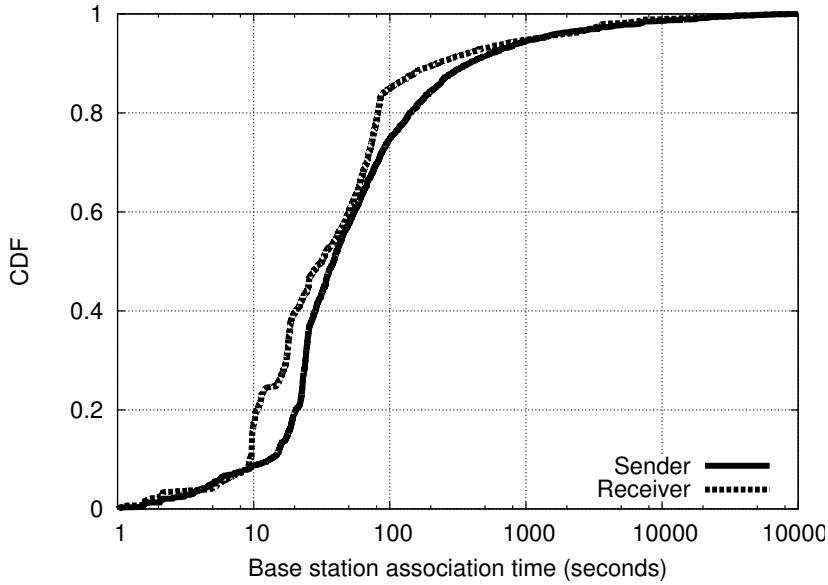
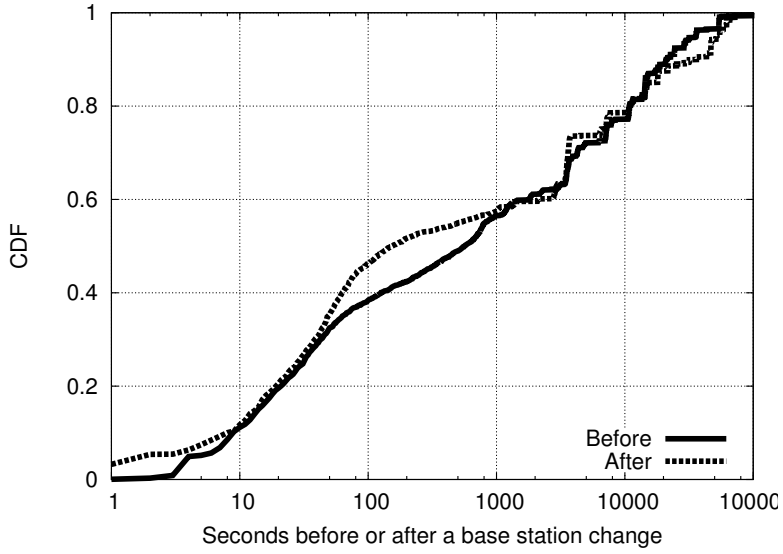
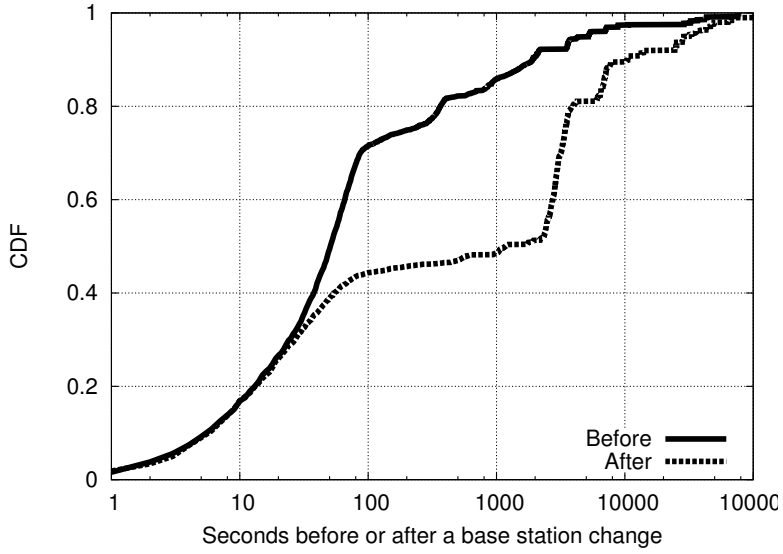


Figure 6.14: CDF of base station association times.

We illustrate the CDF for base station association time for both the sender and receiver in Figure 6.14. In this figure we note that the base station association times for each device are nearly identical.

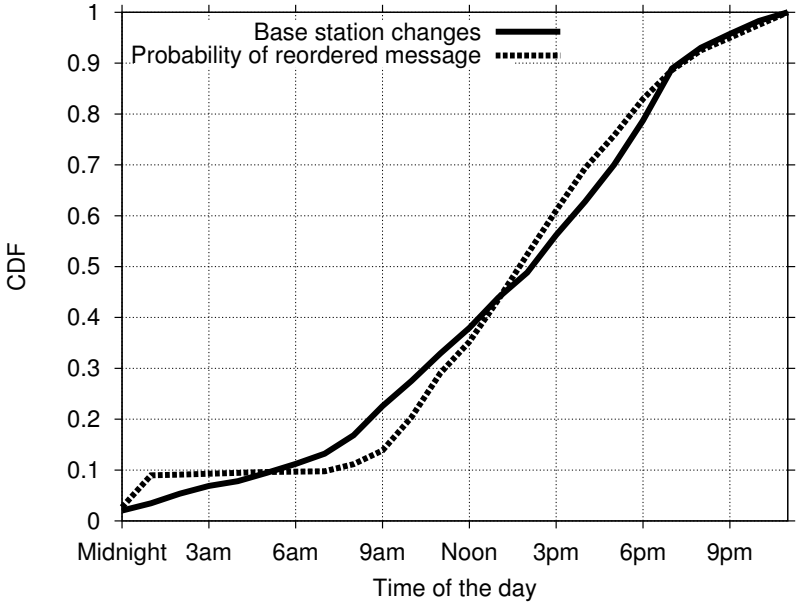


(a) Probability of a reordered message relative to base station change. (Sender)

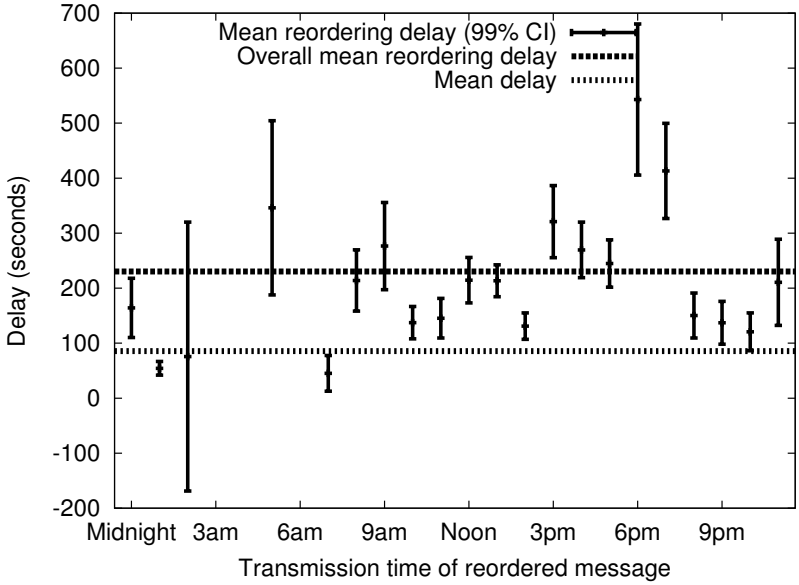


(b) Probability of a reordered message relative to base station change. (Receiver)

Figure 6.15: Analysis of SMS message reordering.



(a) Distribution of base station changes.



(b) Mean delay introduced by reordering.

Figure 6.16: SMS message reordering rate and delay.

On the sending side of our experiment, we found that there was no correlation between base station change and probability of 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 6.15(a).

On the receiving side there is a significant correlation between base station change and probability of message reordering. We observed a dramatic difference between the distributions of reorderings. As illustrated in Figure 6.15(b), nearly 70% of reordered messages are transmitted 100 seconds 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 did observe a close relationship between base station association time and the probability of reordering. Figure 6.14 reveals that nearly 80% of base station associations last less than 100 seconds. However, a closer examination of the probability of reordering 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 as the 'Before' function in Figure 6.15(b). Interestingly, nearly 50% of the messages that are received out of order were received within the 45 seconds prior to a base station change. Despite a rigorous examination of signal strength and base station change patterns, we have no explanation for this observation.

Time-of-day: Finally, we consider the relationship between the delay introduced by message reordering and time-of-day. Recall that for monotonically increasing message IDs, reordering delay is the difference in time between the arrival of a reordered message i and the minimum receive time of message $i+1, i+2, ...$ Over the course of a day, we observed a distinct difference between the frequency of base station changes and the probability of reordered messages. We illustrate the difference in Figure 6.16(a). In this figure, we compare the daily CDF of base station hand-overs with the CDF of reordered messages. The CDF curves illustrate that there is an higher probability of a reordered message than a base station change between the hours of midnight to 3 am and 1:30 pm and 6 pm. This relationship is reversed during the period of 6 am and noon. Although this difference is only minor, we note that 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. Overall, we observed a mean reordering delay of 235.98 seconds.

This delay is in addition to a mean delay of 99.56 seconds. Therefore a reordering introduces a nearly 240% increase in overall delay. Figure 6.16(b) illustrates the 99% CIs (two-tailed t distribution) for each hour of the day. In this figure we immediately see that there is significant difference between the hours of the day. However, by categorizing reordering delay into three periods of the day, midnight to 9 am, 9 am to 6 pm, and 6 pm to midnight, the intra-day variation is more obvious. We found that 13.8% of reordered messages are transmitted between midnight and 9 am, 69.1% are transmitted between the hours of 9 am to 6 pm, and the remaining 17.1% occur between the hours of 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.

6.4.5 Summary of results

We briefly summarize the key results of our characterization. Table 6.2 highlights the key findings of our characterization and how each results impacts the design of the transport protocol. Table 6.3 provides the mean values for each channel characteristic for each test device.

6.5 Transport protocol design

Although we predict that SMS messages will continue to decrease in monetary cost, the service was not yet free at the time of this work. In addition to the design criteria outlined in Table 6.2, 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 are increased by nearly a factor of three and delays can increase by nearly an order of magnitude, nearly 45% of messages are reordered, and the

Key Findings	Design Impact
Transmission time	
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.
Message delay	
Delay is highly correlated with the device 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 can be agnostic to the quantity of messages transmitted.
Devices utilize a dedicated control channel to retrieve multiple messages from the service center. Despite high delays and message reordering, messages are received at a steady rate in batches.	The protocol may assume a relatively constant message arrival rate.
Message loss	
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.	Message delivery is not guaranteed. The protocol must detect and correct missing fragments of data.
Message reordering	
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 cellular signal strength, the ability to programmatically monitor the signal strength of a base station is not available on most mobile platforms [125].

Table 6.2: The impact of SMS channel characteristics on design of a transport protocol.

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 6.3: Summary of device measurements.

loss ranged between 9% to 16%. 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.

6.5.1 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 [168, 179]. Because losses are known to occur in the cellular network [201], applications achieve reliable data transport by utilizing a per-message ack. After sending each message, the sender initiates an 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 avoids of timers at the receiver. This protocol is therefore 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 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 transfer time.

6.5.2 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.

Flow control and error control

SMS-TP provides flow control and error control through a simplified version of the NETBLT protocol [36, 37]. 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 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 transmits messages in a continuous burst. Given that our findings indicate that messages can be transmitted rapidly, this attribute of NETBLT significantly simplifies the complexity of our protocol implementation.

Our approach differs from NETBLT by using only one buffer that is at most 32 KB; thus removing the need to coordinate buffer transfer and to reassemble buffers at the receiver. Like NETBLT, buffers are fragmented into SMS message sized *chunks* for transmission to

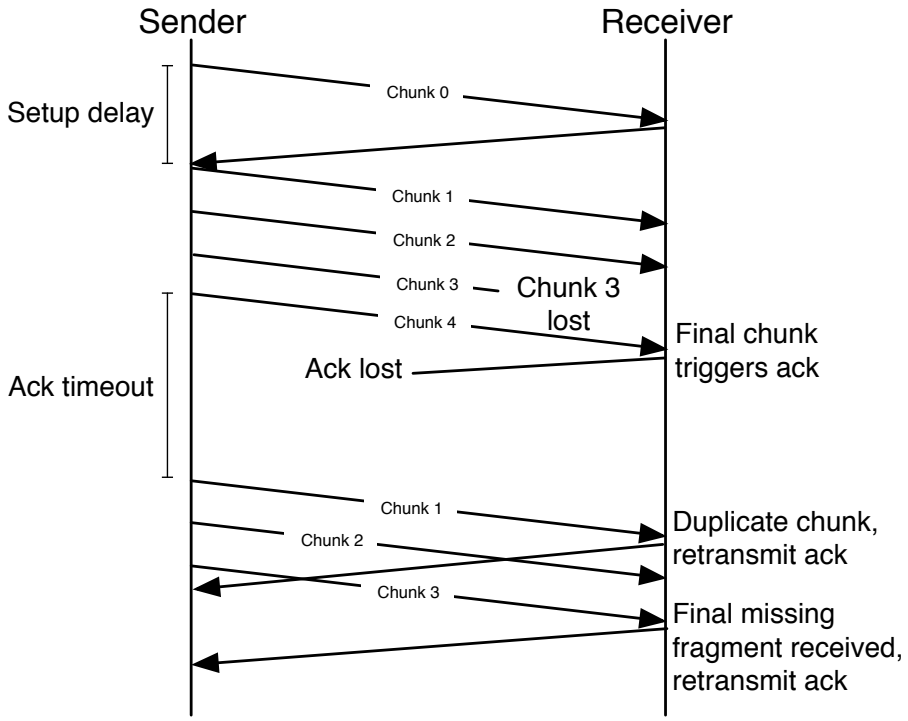


Figure 6.17: Example scenario for acknowledgement timer at the sender.

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 ID, 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 three times before failing and reporting the failure. 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 resending messages; however, the sender is not aware of losses or reordered messages in the network. The sender continues to transmit messages until there is no more data to send. The protocol is illustrated with a sample five-fragment scenario in Figure 6.17.

Algorithm 1 SMS-TP algorithm at the sender.

```
setup_delay = 120 seconds
send initial chunk
time_sent = time_now
set_timer(time_now + setup_delay)
while true do
  event ← waitforevent
  if event = message received then
    record acked chunks {All messages received are acks}
    if all chunks acked then
      exit {Success}
    else
      if event = initial ack then
        setup_delay = time_now − time_sent
      end if
      send all nonacked chunks {All chunks except the initial chunk}
      reset_timer(time_now + setup_delay)
    end if
  else if event = timer expired then
    if retry_count > 3 then
      exit {Failure}
    else
      retry_count = retry_count + 1
      sent all nonacked chunks
      reset_timer(time_now + setup_delay)
    end if
  end if
end while
```

Acknowledgement timers

The sender maintains an ack timer initialized to 120 seconds. Each message sent by the sender causes an ack timer to be reset. Because delay is independent of the transmission index, we know that the first message follows the same delay distribution as subsequent

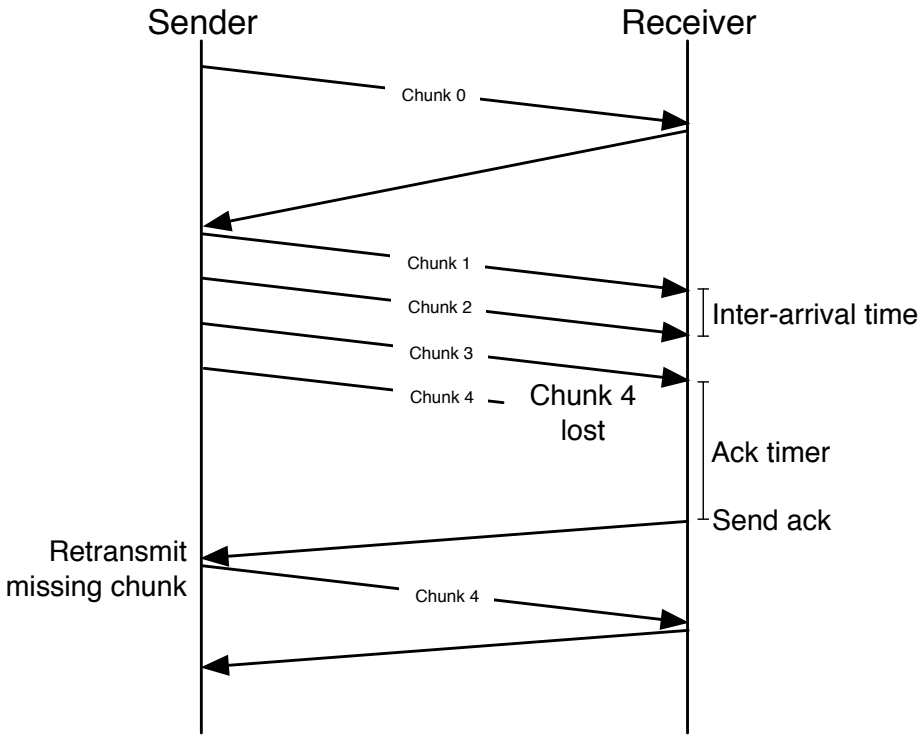


Figure 6.18: Example scenario for acknowledgement timer at the receiver.

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 redundant message retransmissions. We 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. The pseudocode for this algorithm is given in Algorithm 1.

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 message bursts arrive in batches with a steady inter-arrival rate. 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 β times the inter-arrival time. Recall from Figure 6.10 that 98% of messages transmitted back-to-back arrive within

Algorithm 2 SMS-TP algorithm at the receiver.

```
intra = 120 seconds
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 = α(intra) + (1 − α)(time_now − time_recv)
      reset_timer(time_now + intra ∗ β)
    end if
    time_recv = time_now
  else if event = timer expired then
    if retry_count > 3 then
      drop partial data
      exit {Failure}
    else
      send selective ack
      retry_count = retry_count + 1
      reset_timer(time_now + last timer value ∗ γ)
    end if
  end if
end while
```

three times the median inter-arrival time. We therefore consider $\beta = 3$ to be a reasonable selection. Messages that are delayed beyond this threshold are probably lost. Figure 6.18 illustrates the same five-chunk sample scenario where the ack timer is used by the receiver.

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 γ times its previous value. The choice of γ is less critical than the choice of β. 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 2. 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. If a duplicate chunk is received, then the receiver assumes that a previously transmitted ack was lost and immediately transmits an ack representing the current state to the receiver.

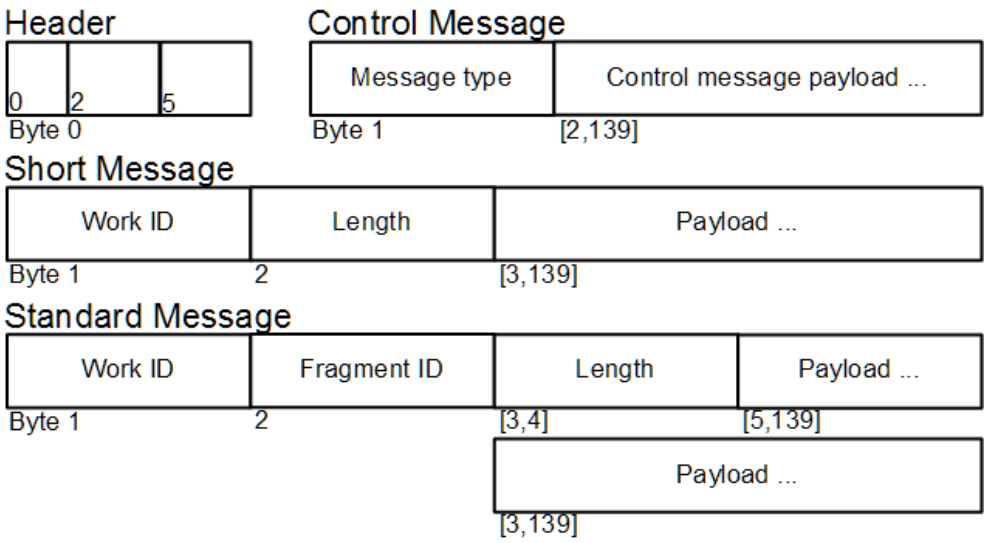


Figure 6.19: SMS-TP message format.

Message format

Each transmitted SMS message contains a small fixed size header and message payload. The first byte consists of a two-bit protocol version, a three-bit message type, and three flag bits. The first flag bit signals that the payload is compressed and the last two bits are currently unused. In the future we foresee using one-bit to signal that the payload is encrypted.

Messages exchanged between two SMS-TPs consist of *short messages*, *standard messages*, and *control messages*. The format of these messages is illustrated in Figure 6.19. Short messages consist of data that is small enough to fit into a single SMS message. We have included this special case to reduce the header size by two bytes. Standard messages are designed to provide fragmentation and reassembly of larger data. These messages consist of a 5-byte header, and a 135-byte payload. We have chosen to limit the maximum data size supported by SMS-TP to 32 KB. We believe that 32 KB (243 SMS messages) is a practical upper bound on data size. Systems that must exchange larger amounts of data may fragment their data into 32 KB sized units or switch to a cellular data service such as GPRS/EDGE. Finally, control messages are used to transport the selective acks described above.

6.5.3 Architecture

The software architecture of SMS-TP , as illustrated in Figure 6.20, 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.

The SMS-TP consists of the following high-level components.

SMS-TP API: The SMS-TP API is the application interface into the SMS-TP. This API allows *host applications* to send and receive data, register a view, configure, and start and stop the SMS-TP. This interface provides five sub-interfaces: the *SMS Data*, *SMS Address*, *SMS Logger*, and *SMS View* interfaces. Receive buffers within the SMS-TP

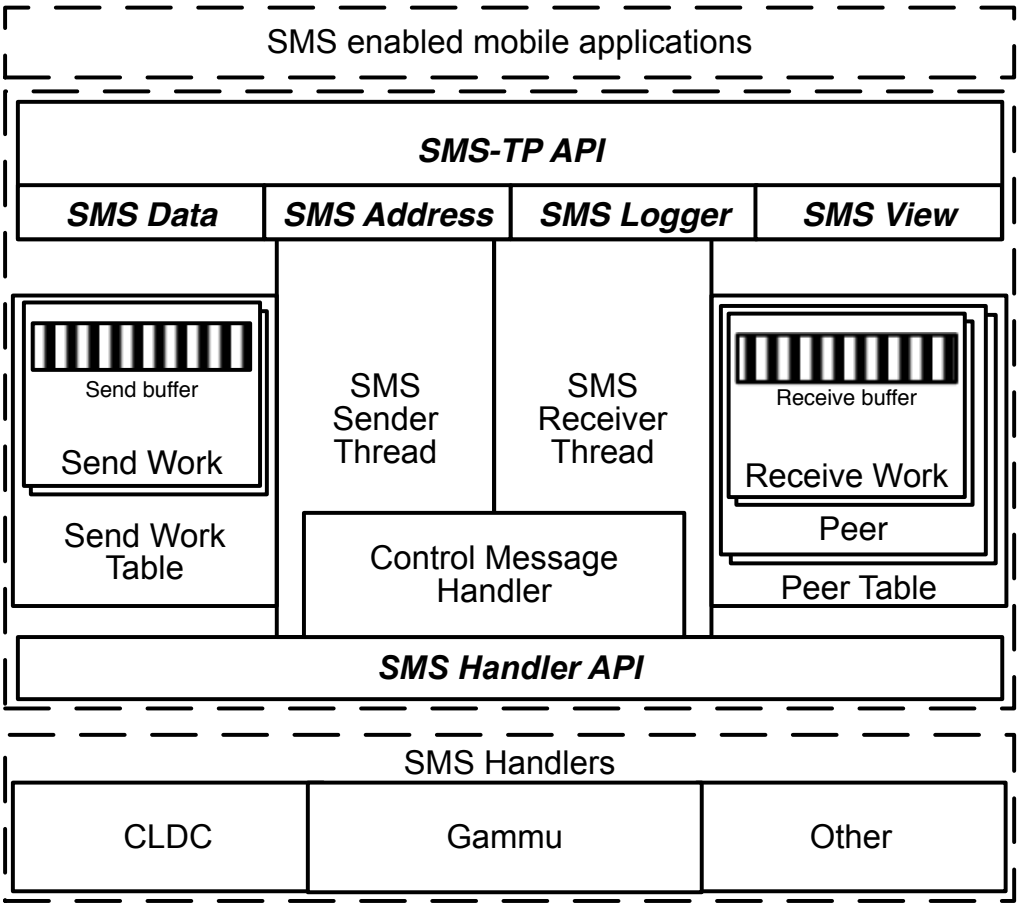


Figure 6.20: Software architecture of SMS-TP including example SMS Handlers.

implement the SMS Data interface. This interface allows fully received data to be passed directly to applications without allocating new memory. The SMS Address interface is designed to ease integration of the SMS-TP into existing applications. An address consists of a phone number used to send an SMS to a device and an optional GSM port number. This interface is implemented by objects in the host application that are responsible for maintaining addresses and removes the need for redundant copying of addresses (typically strings) into the SMS-TP. The SMS Logger is a simple interface that allows the host application to specify how the SMS-TP should log its activities. The SMS View API provides a call back mechanism for the host application to be notified of changes to the

internal state of the SMS-TP. This interface is primarily intended to support a GUI in the host application. More detail on the SMS-TP API is available in our implementation [124].

SMS Handler API: Each mobile platform has a unique method for sending and receiving SMS messages. The SMS Handler API is designed to abstract these differences from the internal operation of the SMS-TP.

SMS Send Work: This component contains the data sent from host application data and is responsible for providing SMS message sized chunks of data to the SMS Sender for transmission. Chunks are retrieved from the work object by a linear sequence number starting from the beginning of the given data. This component also maintains the state of a transaction, which includes the messages that have been successfully received, the last time ack was received or message was sent. The symmetric component is *SMS Receive Work*, which reassembles messages into application data on the receiver's side.

SMS Sender: The SMS sender is one of the two threads in the SMS-TP. The primary function of this component is to dequeue data passed into the SMS-TP for transmission, compress it if configured to do so, wrap the data in an SMS Send Work object, and transmit chunks of the data as SMS messages. The sender stores SMS work objects in a table, which is keyed by a one-byte integer *work identifier* that is chosen in increasing order by the sender to uniquely represent a transaction.

The SMS Sender maintains two sending queues: a high priority queue for control messages and a second queue for sending messages containing application data. Messages are placed in both queues by the sender thread and by the Control Message Handler when processing received acks. The sender is also responsible for recovering from transmission failures. In many cases, this involves simply pushing the SMS message back onto the sending queue.

SMS Receive Work: On the receiving side of the SMS-TP, data is reassembled within an SMS Receive Work object. This component maintains a receive buffer for the full data size and a bitmap of all received messages. When sending an ack, this special message is retrieved from the work object and transmitted to the sender. As mentioned above, this component implements the SMS Data interface, which allows completed data to be passed to the host application unmodified.

SMS Peer: The SMS Peer stores a set of partial Receive Work objects from a specific source address (phone number). This component also temporarily stores the work identifier

of completed work. This prevents new Receive Work objects from being allocated when highly delayed messages arrive after the work has completed. When an SMS Peer contains no work objects or completed work identifiers, it is considered empty and is deleted.

SMS Receiver: The SMS Receiver is the second thread in the SMS-TP. The primary role of this component is to block on the arrival of an SMS message. When a message is received it takes one of two paths: control messages are passed to the Control Message Handler and data messages are passed to an SMS Peer corresponding to the message's source address. If a received message completes a work object, then the work object is removed form its peer container and placed in the application receive queue.

A secondary role of the Receiver is to enforce upper bounds on both the data size and the number of concurrent work objects that can be handled by a receiver. SMS messages that are received for a Receive Work object that is either larger than a user specified limit or exceed 32 KB are discarded.

Control Message Handler: The Control Message Handler implements the communication protocol between two or more SMS-TPs. This component has access to both the sending and receiving data structures and is responsible for handling and dispatching all inter-NIC control messages.

6.5.4 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 [125]. 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 is licenced under the Apache Licence and can be downloaded from our website [124].

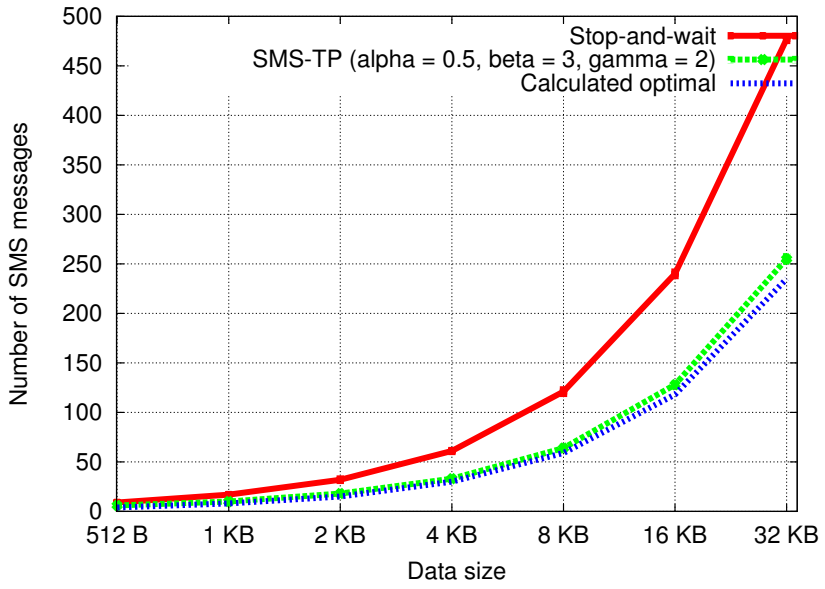


Figure 6.21: SMS message overhead for the SMS-TP vs. stop-and-wait protocol and theoretical optimal.

6.6 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.

6.6.1 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 as in our previous experiments in Section 6.3. 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 repeatedly exchanging data of size 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 data bundle under perfect conditions. We use a constant mean transmission time and delay, with no losses, no reordered 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.

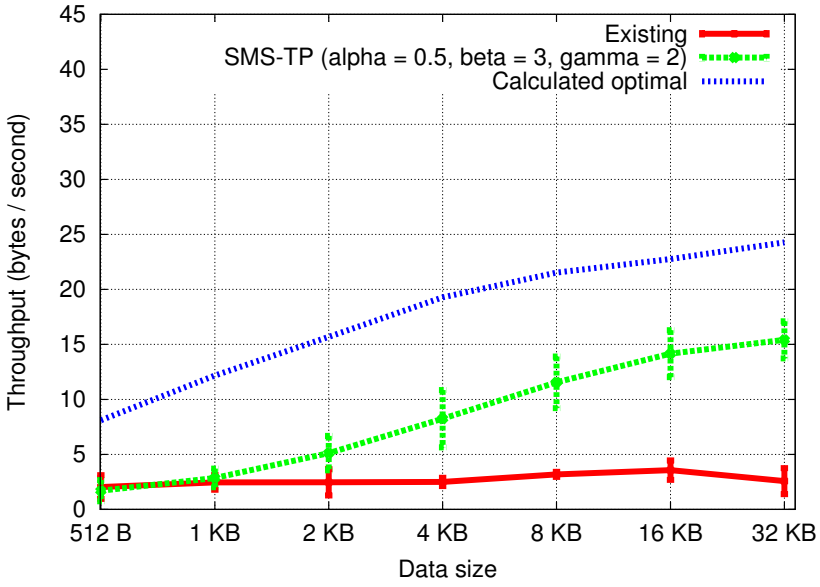
We conduct our evaluation during the hours of midnight to 6 am for consistency in our measurements. Our evaluation is focused on two key variables: message overhead and data throughput. When measuring throughput, we assume that the transaction is over when the receiver finishes writing the data to file,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

Message overhead

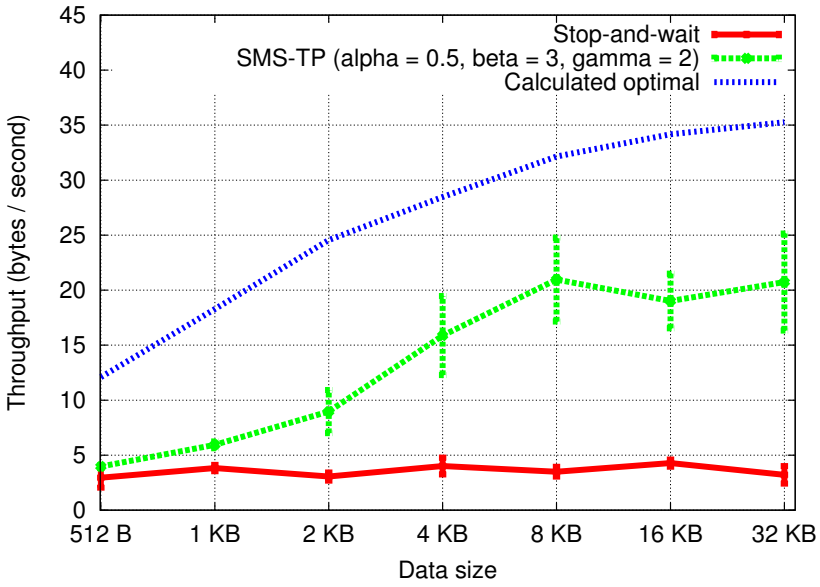
The message overhead (99% CI) for transferring each data size for each of the three cases using the 8820 is illustrated in Figure 6.21. We found that the overhead using the cell phone was statistically equivalent. As expected, we found that SMS-TP offered a nearly 50% gain over the existing method. This gain is due to our consolidation of acks 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 worst case, the message overhead of SMS-TP was only 8.5% below the optimal case.

Throughput

We found that SMS-TP yields a significant improvement in data throughput over the existing method. We illustrate the results of our evaluation for both the cell phone and



(a) SMS-TP throughput using a tethered cell phone.



(b) SMS-TP throughput using a smartphone.

Figure 6.22: Throughput analysis of SMS-TP.

smartphone in Figure 6.22(a) and Figure 6.22(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 KB or more of data. Using the smartphones, our protocol improves throughput over the stop-and-wait protocol by as much as 545%! The significant gap in throughput between the cell phones and smartphones is primarily due to communication failures with the cell phone that occurred approximately 5% to 10% of the time and inhibited the phone's ability to both send and receive messages and deliver messages to the PC over the serial connection. We expect that in the absence of the communication failures, data throughput using a tethered cell phone would improve dramatically.

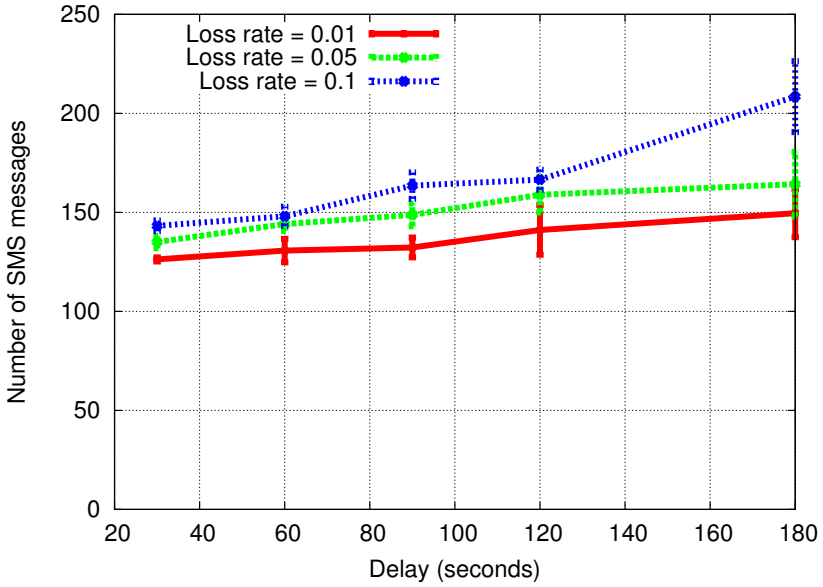
We also found that in the worse case SMS-TP performs worse than the optimal case by approximately 40%. Given the generosity of the assumptions made in the optimal case, we believe that this is a good first step. Narrowing this gap (if possible) will be an area of future study.

6.6.2 Simulation

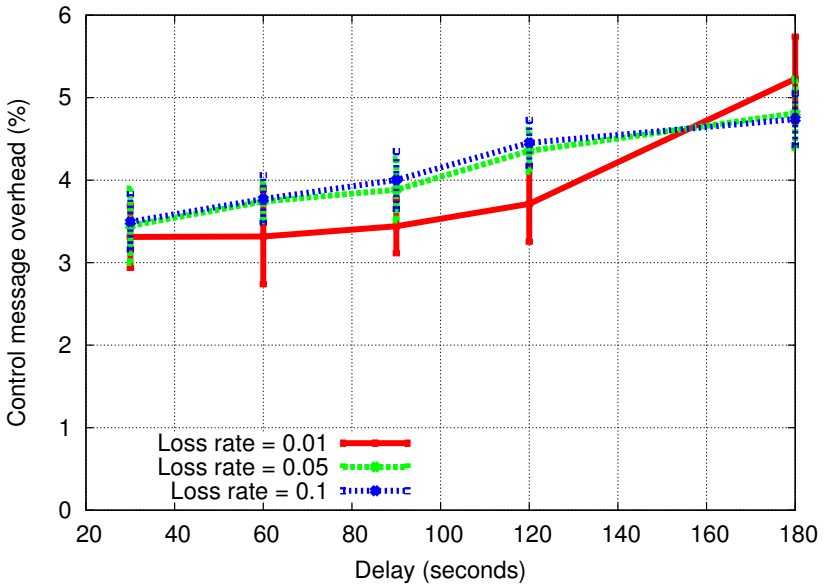
In the second phase of our study we examine the effect of loss rate and delay on the message overhead and the throughput of SMS-TP through simulation. We select these characteristics from within the 90 percentile of values observed by Zerfos in [201]. Our simulation therefore examines the performance 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 at 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 [124].

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.



(a) SMS message overhead for SMS-TP for variable long rate and delay.



(b) Control message overhead of the SMS-TP vs. delay.

Figure 6.23: SMS message overhead for the SMS-TP.

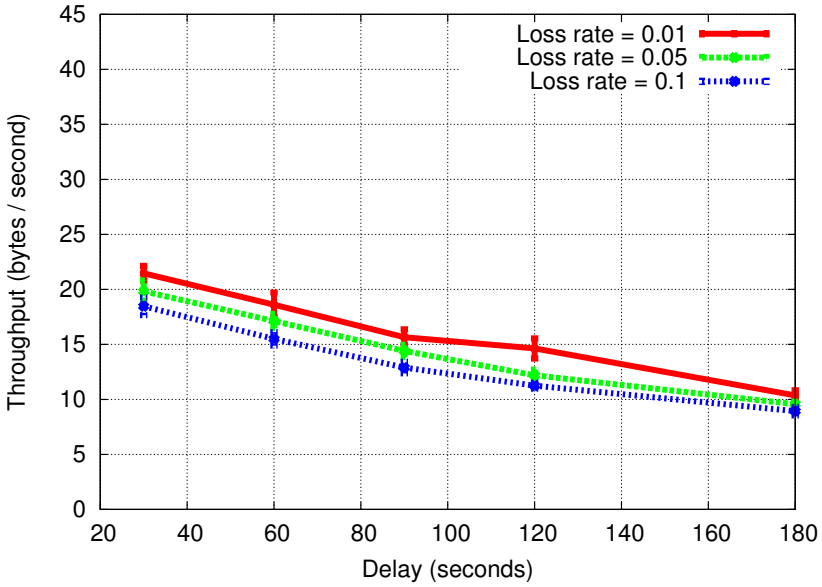


Figure 6.24: Throughput for SMS-TP for variable message loss rates vs. delay.

Message overhead

The mean total message overhead is illustrated in Figure 6.23(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 6.23(b).

Throughput

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 6.24. 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 affect 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.

6.7 Chapter summary

This chapter has presented the 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 characterises 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 device used, time-of-day, signal strength, and base station changes. This is the first work 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.

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.

While the SMS-TP is clearly an invaluable component of our CTN system, this work can have impact anywhere that 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 KioskNet system [66] and to ex-

change cryptographic information between mobile devices in NearbyFriend [207]. We believe that many other mobile systems could benefit from this work.

Chapter 7

An Empirical Approach to Smartphone Energy Consumption

The last chapter introduced an optional SMS-based control channel that can provide end-to-end connectivity between non-collocated device. This chapter introduces a second optional component, the Energy Management Oracle, that serves as a fail-safe mechanism to prevent devices participating in a CTN from depleting their battery. By querying the oracle before executing an energy intensive operation, the prototype CTN application can adapt its execution to preserve battery life. The design of the oracle is influence by a large-scale user study, and evaluated using a custom-built experiment platform.

7.1 Introduction

The proliferation of smartphones is driving a rapid growth in mobile applications. The design of these applications is governed by several constraints that are unique to mobile computing environments. Of these constraints, energy is the one resource that when depleted will render all of the applications on the mobile device, including emergency and essential applications such as the phone, inoperable. Unfortunately, while global demand and use of mobile applications continue to expand, the energy density of smartphone batteries has grown at a comparably insignificant rate [180]. Energy is therefore a critical resource that must be carefully considered in the design of mobile applications.

CTNs are among a class of applications that must operate with little or no user-intervention, must operate over long durations, and consume large amounts of energy. Delay tolerant networking applications, such as those in References [13, 113, 144, 175], are also examples of this class of application. Each application must forward data bundles autonomously and operate continuously since being offline prevents data exchange with neighbouring devices. These applications also consume large amounts of energy due to their use of wireless network and file I/O during opportunistic connections, and can easily deplete a battery if left unchecked. We are therefore faced with the question of how to maximize benefit to the user while minimizing impact.

We approach this complex question through a large-scale user study that examines the charging characteristics of smartphone users. Using this information, we build the tools and libraries that allow mobile application developers to adapt to increasing energy constraints. The contributions of this work are threefold:

- We build a dataset containing the smartphone usage and energy consumption characteristics of 20,100 BlackBerry smartphone users. Our dataset contains over 1150 years of cumulative data from users spanning 23 time zones and every BlackBerry device-type released since early 2006.

- We exploit user energy traces in our dataset to build the *Energy Emulation Toolkit* (EET). Using this tool, developers of energy intensive applications can test their applications' energy consumption behaviour against existing energy traces. Developers may then alter their design or tune algorithm parameters to reduce energy consumption and ensure that their application does not deplete users' batteries.

- By classifying users into one of three groups according to their unique energy consumption characteristics, we demonstrate that energy level can be predicted to within 7% error within an hour and within 28% error a full day in advance. Using our prediction algorithm, we build the *Energy Management Oracle* (EMO) library. By querying the EMO before executing an energy intensive operation, applications can adapt their execution to achieve a near optimal successful execution rate.

We begin this chapter with a summary of related work. We then present an overview of our usage study and the dataset that we have built. Section 7.4 presents the design and evaluation of the EET tool. Section 7.5 describes our user classification scheme, our energy level predictor, and the EMO library. We summarize this chapter in Section 7.6.

7.2 Related work

Although energy consumption has been widely studied [12, 95, 96, 121, 138, 149, 167], little is known about the charging characteristics of users. Rahmati et al. were the first to qualitatively and quantitatively assess how users consumed energy. They provide preliminary insight into charging behaviour, user interfaces for power-saving settings, user knowledge, and user reaction to battery levels [147]. Their experiment consisted of ten users using unfamiliar devices over a one-month period. Their experiment driver also reduced the life of the device to about 40% of its usual lifetime. Falaki et al. have measured the energy consumption and application usage behaviour of 33 Android and 222 Windows

Mobile devices [52]. Our BlackBerry user study complements Falaki's work; however, our dataset is approximately two orders of magnitude larger. Moreover, all of our 20,100 users are running our experiment software on their *personal* devices for periods up to several months in duration. We believe that our work is a more representative study. Finally, our experiment driver is event driven, does not record data to flash memory, and consumes less than 1% of additional battery life. Its operation is transparent to the user.

Cignetti et al. present the design of an energy model that estimates energy consumption with respect to the actions performed in an application [34]. Given that today's mobile devices are highly concurrent and that energy intensive tasks may take place without user intervention, actions performed by users are insufficient indicators of future energy levels. Bellosa examines OS-directed power management in Reference [21]. In this work, system activity is throttled to achieve a desired energy consumption goal. This work is practical for homogeneous server workloads, but would not work under the heterogeneous workloads of a personal computing environment. Moreover, throttling of any component would be noticeable to the user and could adversely affect user experience. This approach also requires knowledge of the energy consumption rates of the underlying hardware to accurately throttle the energy consumption rate of the system. Ravi et al. exploit a user's location to predict when the user will charge their device [149]. Based on this prediction and the current energy consumption rate, their work determines the portion of the battery that can be used by other applications and warns the user of potential battery depletion.

7.3 Smartphone usage study

Our measurement study is focused on understanding how smartphone users interact with their devices and how they consume and replenish energy. We begin with a brief overview of the measurement study driver and data that we collect. We then discuss the primary challenges in implementing the driver and producing a large-scale dataset of this kind.

7.3.1 Measurement driver

Most user-centric experiments distribute smartphones to a mere 10-30 users: fellow researchers, staff members, or undergraduates. Although this has been a convention for

the past several years of mobile computing research (primarily due to convenience), this method of experimentation has several significant flaws:

- Experiment-enabled devices are typically given to participants on a temporary basis, which can affect users' interaction with the device and degrade the accuracy of user-centric results.
- Participants are often unwilling to use an experiment-enabled device as their primary/personal device. This can further bias the accuracy of an experiment.
- Participants are often intimately aware of their role in an experiment, which may further bias the results either positively or negatively. Participants may, for example, interact with or charge the device differently if they know that their actions are being recorded.

Our measurement study is shaped by three high-level objectives. First, to achieve a high degree of accuracy, we require more participants than previous studies [52, 147]. Second, to obtain many participants our measurement driver must run on a heterogeneous set of mobile devices with a diverse set of capabilities. Finally, our driver runs autonomously on participants' personal mobile devices and must therefore be production-grade software.

We develop two BlackBerry applications to collect the desired data: the Standard Logger and the Background Logger.

Standard Logger

The Standard Logger is an event-driven BlackBerry application that runs continuously in the background of a device. Upon installation, the logger records the following information:

- **Backlight activity**: we infer user-interaction with a device by exploiting the device's LCD backlight being on as an indication of user interaction. Using callbacks from the OS, the logger records the time that the backlight turns on and off.
- **Idle counter**: the OS maintains a counter since the last user gesture. The counter is reset whenever the user presses a button, touches the screen, or moves a trackball

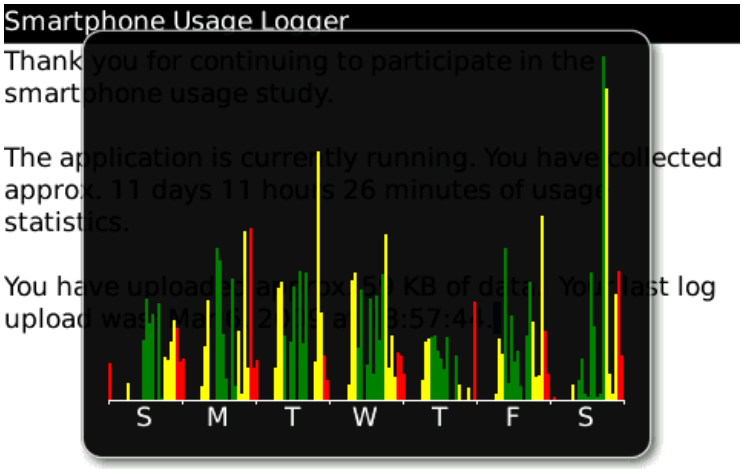


Figure 7.1: Screen capture of the Standard Logger.

on the device. The logger records the counter's value when the backlight turns off to determine the duration that the backlight is on but the user is not interacting with the device.

- **Charging activity**: users' battery charging behaviour can be determined by recording when a device is plugged and unplugged from an external power source such as a USB cable or power adapter.
- **Battery level**: the logger records the battery level every ten minutes to determine how users consume energy throughout the day.
- **Soft shutdown**: the logger records signals that the device is powering on and off to account for inconsistencies in the data. For example, a user may power off a device when the battery is low, plug it in, unplug it at a later time, and power on at a full charge.
- **Device-type**: to differentiate BlackBerry devices we record the device-type and OS version.

The logger contains a UI that allows study participants to observe aggregate statistics and a histogram of their usage pattern throughout the week. Figure 7.1 illustrates a screen

capture of the Standard Logger. Through a menu users may upload the data manually or authorize automatic weekly uploads to a collection server on the Internet.

Background Logger

The Standard Logger alone was not capable of achieving our desired scale. We therefore partnered with a major BlackBerry software developer to augment their existing software quality logging tools with a simplified version of the Standard Logger. We refer to this augmentation as the Background Logger. The Background Logger collects the same information as the Standard Logger and uploads the data to the company's servers each week. We will refer to both loggers generically as the *Logger* in the remainder of this chapter.

Assumptions

The Logger makes one fundamental assumption: that the user does not run applications that programmatically enable and disable the device's backlight while resetting the device's idle time by simulating user input (keystroke injections). Given that application developers rarely use both operations, we consider these to be reasonable assumptions in our study; however, we do take steps to detect and discard invalid data.

7.3.2 Challenges

Achieving our three high-level experiment objectives was challenging. We believe that other mobile researchers can benefit from our insight on the following top four challenges.

Volatile file systems

Logging data to a file is a standard technique to ensure persistent storage in the presence of power failures or other interruptions. However, file systems can be unmounted at any time either manually by the user or automatically when connected to a PC. Adapting to this problem by frequently flushing I/O buffers to persistent storage can have a significant impact on battery life. The Logger therefore exploits two trends among BlackBerry users to provide reliable logging and save energy: users maintain a high battery charge level and

rarely fully power off their devices. Our technique buffers log data in volatile memory until uploading it to our collection server each night.

Energy constraints

An application that has a noticeable impact on energy consumption will not be successful. In preliminary experimentation, we found that users were intimately aware of their device's normal battery life; any noticeable or perceived decrease in battery life would be attributed to the experimental application. As observed in previous studies, polling a device's state can have a detrimental impact on battery life [147]. The Logger exploits callbacks from the operating system whenever possible to reduce the energy cost. Unfortunately, an event driven model can produce inconsistent sequences of events. For example, when powering off a BlackBerry, we frequently did not receive an event signaling that the screen was off until after powering the device on. Similar scenarios were observed when charging. When analyzing the output of an event driven application, one must take into account possible race-conditions in the underlying OS.

Third-party application intervention

User-centric studies must take into account intervention by third party applications. For example, an application that records a user's battery consumption can be affected by a spyware application that is accidentally installed on the device [202]. Unfortunately, collecting a manifest of all applications present on the mobile device would be in violation of the user's privacy. Instead, during the analysis phase of our study, we checked for statistical abnormalities, such as increased energy consumption or excessively long activity blocks. When an abnormality was found, the data was flagged for review, manually inspected, and discarded if suspected to be fraudulent.

Non-linear time

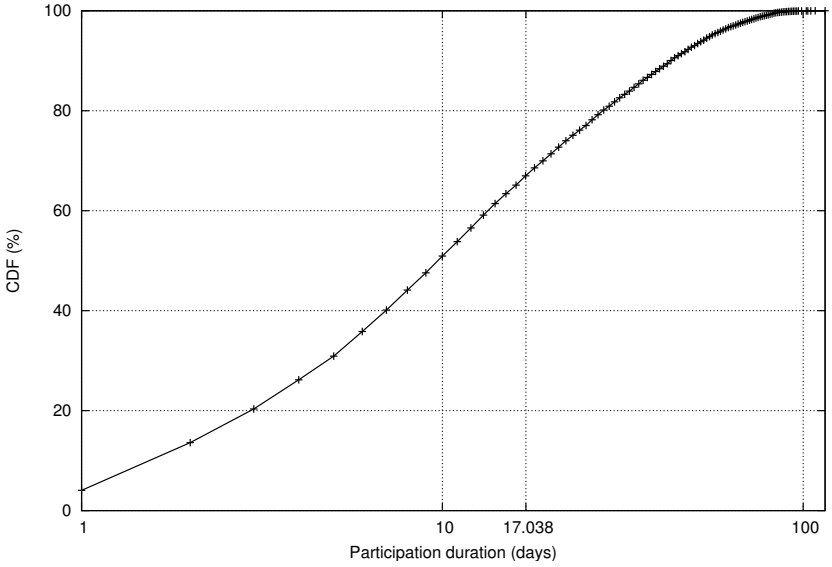
Time synchronization, a task often taken for granted, is the most challenging problem to overcome in autonomous logging. A device's clock can be updated through three means. Plugging a device into a personal computer and synchronizing with a 'device manager' can

change the clock. Users can manually change the time or time zone at any time. A device may also synchronize its clock with the timestamp broadcasted by the cellular network. Changes to the device clock manifest as the appearance of non-linear time. Unfortunately, there is no way to programmatically detect changes to the device clock. We mitigate errors introduced by time changes by detecting when the device is connected to a computer and analyze these events separately from non-connected events. In a large-scale study, users can also traverse many different time zones. UTC time should be used to provide the absolute time of an event. Experiments must also record the device's time zone and when a time zone change takes place.

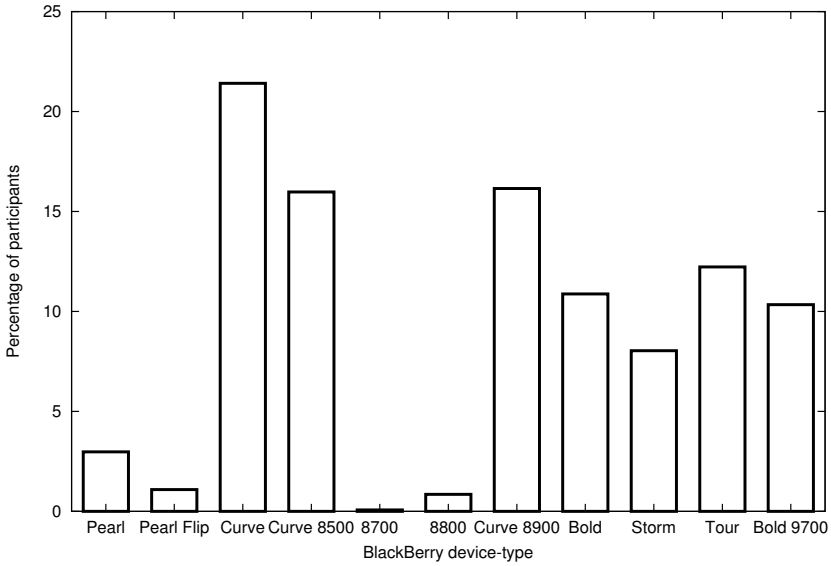
Malicious users

Malicious users are a unfortunate side effect of large-scale studies. If left unchecked, malicious activities can significantly bias the accuracy of an experiment. Over the course of our study, approximately 3.8 years worth of data from the Standard Logger was discarded due to a variety of malicious activities: file manipulation, programmatic manipulation, and simulated user attacks. File manipulation attacks are the easiest to detect. These attacks involve modifying the contents of files, renaming/copying files, or deleting files, and can be countered by storing a hash of the contents of each file and through a sequential file naming scheme. Programmatic attacks involve a third party application that manipulates the device to counter the experimental application. In this case, some users installed an application that randomly enabled/disabled the device's backlight 24 hours a day, 7 days a week, and simulate keystrokes to reset the device's idle time. This attack was detected by visual inspection of the user's weekly usage pattern.

Simulated user attacks are the hardest to detect. These attacks involve downloading experimental software onto a simulated device(s) and then replaying an existing usage log or running a programmatic manipulation attack. This attack was detected by recording the IP address of each set of uploaded data. Multiple uploads from a single IP address are flagged for review. These types of attacks are highly dependent on the experiment parameters. The best approach to detect malicious users and mitigate negative results is to analyze data early and analyze data often - looking for abnormal patterns in every participant's data.



(a) CDF of user participation duration.



(b) Distribution of participating BlackBerry devices.

Figure 7.2: Aggregate dataset statistics.

Software credibility

Gaining the trust of participants is the most difficult challenge of all. Although most users have little technical background, they are acutely aware of the consequences of faulty software on their personal phone[1]. Experimental software designed for users' personal devices must be production-grade software, carefully planned, and bug free. Errors or failures that do exist must be perfectly hidden from the user and have a bounded impact on the user's use of the device. For example, in an early version of the usage logger, when the device associated with a Wi-Fi network, it would flush the current I/O buffer, close the current logging file handle, and attempt to upload the data. If the connection failed, it would retry every minute while the device was within Wi-Fi connectivity. Unfortunately, if the device was associated with a Wi-Fi network but did not have access to the Internet, the logger would produce thousands of small files on the device.

The application, once installed, should startup automatically and require no user intervention. Moreover, the user should not be aware that of the application's presence. However, the application must provide a simple user interface that exposes critical information to the user, such as the amount of data that has been collected and uploaded. If possible, the application should provide some incentive to continue participating in the study. Figure 7.1 illustrates one example.

7.3.3 Aggregate summary

Over six months of data collection, we constructed a dataset that consists of 1150 years of cumulative interaction and energy consumption behaviour from over 20,100 smartphone users. Approximately 15 years of suspicious data from 213 users (a significant quantity of data in its own right) was discarded due to the reasons previously discussed. The CDF of users' participation duration illustrated in Figure 7.2(a). Our participants span 23 time zones and all BlackBerry device-types released since early 2006 [6]. These devices span a wide range of hardware characteristics and similar devices can be found from other manufactures. The proportion of logs collected for each BlackBerry device-type is illustrated in Figure 7.2(b). Our dataset is approximately two orders of magnitude larger than any previous work.

[1]This statement is based purely on anecdotal evidence, but I believe it is an easy argument to make.

7.4 Successful execution prediction

Using energy consumption traces stored within our dataset, this section examines how knowledge of user charging characteristics can improve the design and implementation of energy intensive applications. We begin with an overview of the Energy Emulation Toolkit (EET) followed by an evaluation using a simple sample application.

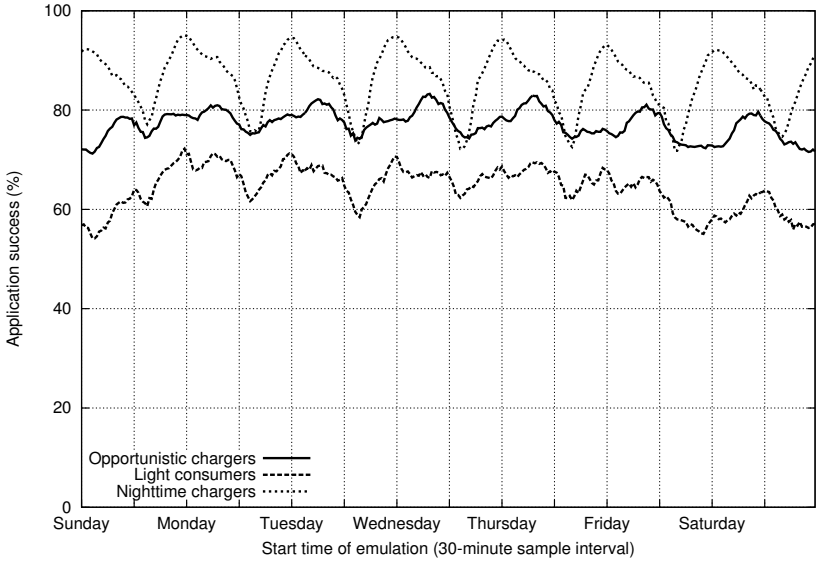
7.4.1 Energy Emulation Toolkit

The EET is designed to predict the *successful execution rate* of energy intensive applications given an energy consumption requirement and a specified duration of execution. An application is considered to have 'successfully executed' if it is able to run to completion without draining a device's battery.

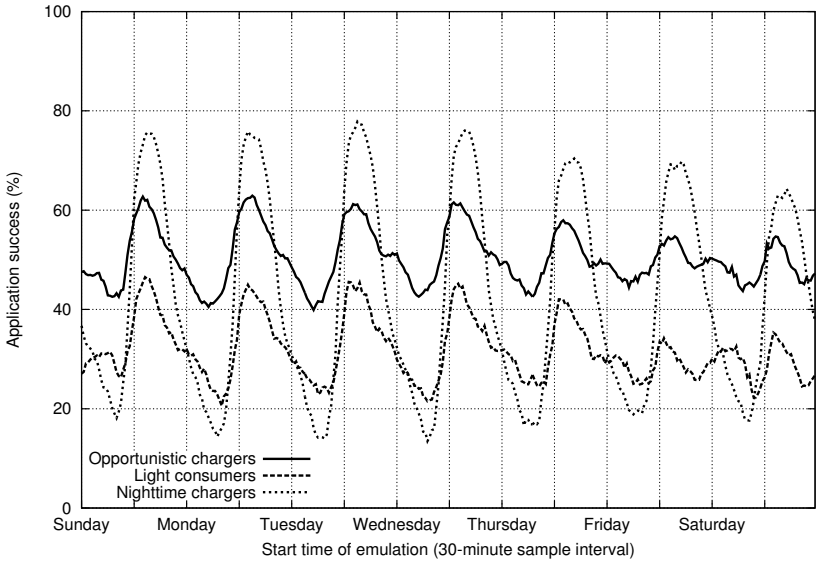
The EET operates by iterating over all smartphone users in a target population of either: all users, users with a specific device-type, or users within a specific *user-type*. We will examine the characteristics of each user-type in the next section. For each user, the EET assembles their energy traces as a linear timeline and begins stepping through in five-minute intervals. Through each step of the timeline, the EET initiates a test to determine if an application invoked at the current time, t, with the specified requirements would succeed. A value of 1 or 0 is recorded for the 30-minute bucket corresponding to t's position in the week to represent a successful or failed execution. The successful execution rate of an application can therefore be provided as a function of the starting time or as an aggregate mean rate.

Implementation

We implement the EET as a stand-alone PERL module that can be imported into any existing PERL program/script. The EET module is initialized with the root directory of the dataset and invoked using a single function call that accepts the energy consumption requirements, the duration of execution, and a value that indicates the target population. Because the EET scans energy trace files linearly it accumulates little state during execution; the EET's runtime footprint when traversing the entire dataset is approximately 46 MB.



(a) Six-hour execution duration (569 mW over 6 hours).



(b) Twelve-hour execution duration (569 mW over 12 hours).

Figure 7.3: Successful execution rate by user-type.

Application	Duration	Rate
GSM phone call (900 mW)	1 hour	95.2%
Video playback (1.1 W)	2 hour	84.3%
Bluetooth Class 2 (idle) (2.5 mW)	24 hours	100%
Bluetooth scan (1 minute scan period) (75 mW)	24 hours	100%

Table 7.1: Sample successful execution rates.

7.4.2 Evaluation

The EET can be applied to both simple and complex applications. Table 7.1 provides examples of simple applications and their mean successful execution rate.

We demonstrate the use of the EET through a simple application that mimics the behaviour of our prototype CTN application under a very simple, predefined workload. This application consists of a thread that scans for neighbouring Bluetooth devices every one minute. On every second scan, the application connects to a server over Wi-Fi and uploads 100 KB of memory-resident data. This process mimics a metadata exchange. Once per hour the application reads a 100 MB file from flash memory and uploads the contents to the server over Wi-Fi. It then downloads 100 MB of data from the server and subsequently writes the data to persistent flash memory. This process emulates sending and receiving content. We implement the simple application, run it on a testbed of BlackBerry devices, and measure the energy consumed. Assuming that the energy consumed by the hourly network and flash memory I/O is amortized over the full hour and found that the average consumption rate is 569 mW. We illustrate the successful execution rate for the sample application over a six-hour and twelve-hour duration for each user-type in Figure 7.3(a) and Figure 7.3(b) respectively. We will not analyze these sample figures because the results for each user-type will be obvious to the reader by the end of Section 7.5.3.

7.5 Energy prediction

The EET can provide developers useful insight into the expected behaviour of the application when deployed on real devices. Using the EET, developers can alter their software design or tune parameters to reduce energy consumption and the likelihood that the appli-

cation will deplete the user's battery. Unfortunately, the EET cannot benefit applications whose energy consumption rate is not known *a priori*. We address this problem using the EMO library in Section 7.5.5.

We begin our design of the EMO by first describing the high-level energy characteristics of a device. We then describe our user classification method in Section 7.5.2. We outline three identified user-types in Section 7.5.3, and present an analysis of our user classification based energy level predictor in Section 7.5.4. We conclude this section with an analysis of the EMO library.

7.5.1 Energy characteristics of a device

Charge and discharge durations: These are the durations that a device is plugged into or unplugged from an external power source. This property has a direct impact on a device's battery life. Devices that charge for longer durations could have a higher expected battery level than those that do not. Similarly, devices that discharge for long durations could have a battery level that is lower than expected.

Charge initiation time/level: The charge initiation time is the time of day when the user begins to charge their device. Regularity in charge initiation time could be used to infer when energy is likely to be replenished. A relationship between charge initiation and battery level could also be used to predict battery level.

Battery level: Patterns in battery level over the course of a day or week could be an ideal parameter for predicting future battery life.

Charge and discharge rates: These rates are the percentage of total battery capacity that is replenished or depleted while plugged or unplugged from a power source. We observed a significant correlation between the discharge rate, device-type, and the time of day. This result is not surprising considering the energy consumption disparity between a device sitting idle at night and a device that is used frequently throughout the day.

Other characteristics: Although our dataset contains a rich body of smartphone interaction data, we found that there was little correlation between energy consumption and user interaction.

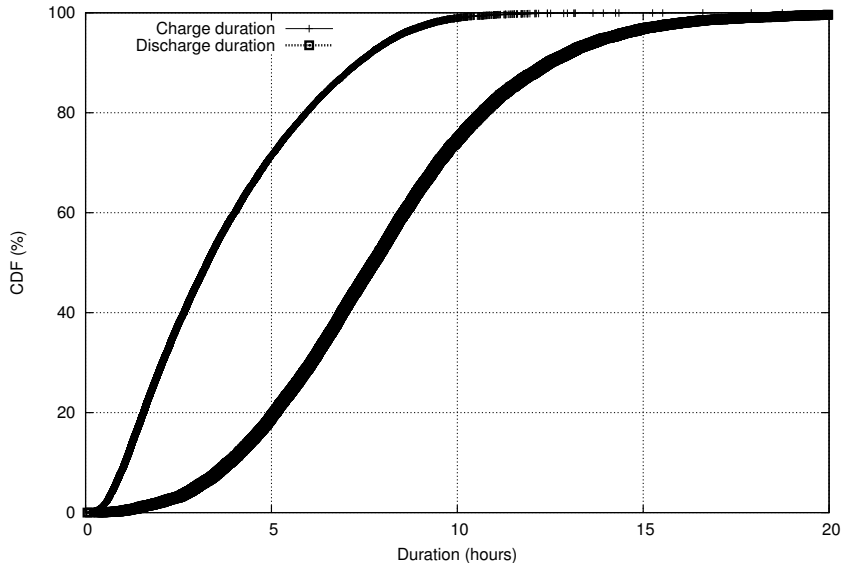


Figure 7.4: CDF of participants' mean charge and discharge duration.

7.5.2 Classification method

Our user classification method is an iterative process to determine the subset of high-level energy characteristics that differentiate users based on their predictive ability. On each iteration we cluster users using the selected high-level energy characteristics. We then use the mean prediction error derived from the clustering to evaluate the selected characteristics. Given that the choice of prediction algorithm is dependent on the choice of input parameters, a circular dependency exists. We therefore begin our process by defining a baseline prediction algorithm.

Baseline prediction algorithm

The design of our baseline prediction algorithm was shaped by the high-level goal of providing predictions over long durations that may extend over multiple charge and discharge cycles. The prediction algorithm must therefore adapt to three dominant trends identified during an aggregate analysis of our dataset. First, the charge and discharge durations vary widely among users. The CDFs of both the mean charge and discharge duration are

Algorithm 3 Predict(t_{curr}, b_{curr}, t_{last}, t_{pred}, γ)

Require: $t_{last} \leq t_{curr}$, $t_{curr} < t_{predict}$

```
b_pred ← b_curr
while t_curr < t_predict do
  if γ = CHARGING then
    t_duration ← δ_charge[bucket(t_last)] {Retrieve expected charge cycle duration.}
  else if γ = DISCHARGING then
    t_duration ← δ_discharge[bucket(t_last)] {Retrieve expected discharge cycle duration.}
  end if
  if t_last < t_curr then
    {Algorithm invoked mid-cycle}
    t_duration ← t_duration − (t_curr − t_last)
  end if
  {Iterate through each bucket over the charge or discharge duration.}
  for bucket β ∈ t_duration do
    if γ = CHARGING then
      b_pred ← min(100, b_pred + ρ_charge[β] ∗ |β|) {Increment the predicted battery level.}
    else if γ = DISCHARGING then
      b_pred ← max(0, b_pred − ρ_discharge[β] ∗ |β|) {Decrement the predicted battery level.}
    end if
  end for
  t_curr ← t_curr + t_duration {Advance time by duration of cycle.}
  t_last ← t_curr {The end of a cycle is the beginning of another.}
  γ ← ∄γ {Alternate the charging state between CHARGING and DISCHARGING.}
end while
return b_pred
```

illustrated in Figure 7.4. These durations vary throughout the week. Second, the mean discharge rate varies significantly over the course of day: a device sitting idle at night consumes energy at a lower rate than a device that is active throughout the day. Moreover, there is a large disparity between the mean charge rate (46.8%/hour) and the mean discharge rate (4.2%/hour).

Our prediction algorithm is specified formally in Algorithm 3. We sub-divide each

Algorithm 4 bucket(t_{curr})

$bucket\_size = 1800$ {30 minutes}
$day = get\_day\_of\_week(t_{curr})$ {Where $0 \geq day < 7$}
$hour = get\_hour\_of\_day(t_{curr})$ {Where $0 \geq hour < 24$}
$second = get\_second\_of\_hour(t_{curr})$ {Where $0 \geq minute < 3600$}
return $day * 86400/bucket\_size + hour * 3600/bucket\_size + \lfloor second/bucket\_size \rfloor$

week into 336 discrete 30-minute buckets and initialize the algorithm with four vector parameters: δ_{charge}, $\delta_{discharge}$, ρ_{charge} and $\rho_{discharge}$. The δ vector contain the mean charge and discharge durations for cycles initiated during a specific bucket. If δ is undefined in a given bucket, then the algorithm uses the mean value of the target population. The ρ vector contains the mean charge and discharge rate for each bucket. Again, if a value within ρ is undefined then the mean charge and discharge rate over the target population is used. Each invocation of the algorithm requires the following parameters: the current charge or discharge state (γ), the time that the current charge or discharge cycle (t_{last}) began, the current time (t_{curr}), the current battery level (b_{curr}), and the desired prediction time (t_{pred}). Using the time that the current charge or discharge cycle began, the algorithm examines δ to determine the expected durations of the cycle and the time that the next cycle will begin. While stepping through each cycle, our algorithm increments (charge) or decrements (discharge) the predicted battery level (b_{pred}) at the rate specified in ρ for the duration specified in δ.

Clustering strategy

With a baseline prediction algorithm we now evaluate the prediction accuracy derived from different user features. This iterative process involves selecting a subset of energy characteristics to cluster users and applying the prediction algorithm to each user-type cluster. Each user point is placed in an n dimensional space using the mean value of each of the n characteristics selected. At the end of each iteration we evaluate the gain/loss in prediction accuracy and select a new subset of characteristics.

Clustering was performed using the k-means algorithm with the mean Euclidean distance between variables as our distance metric. In each iteration we loop through the values $k = [2, 6]$ clusters. The prediction algorithm is applied to each user clustering for

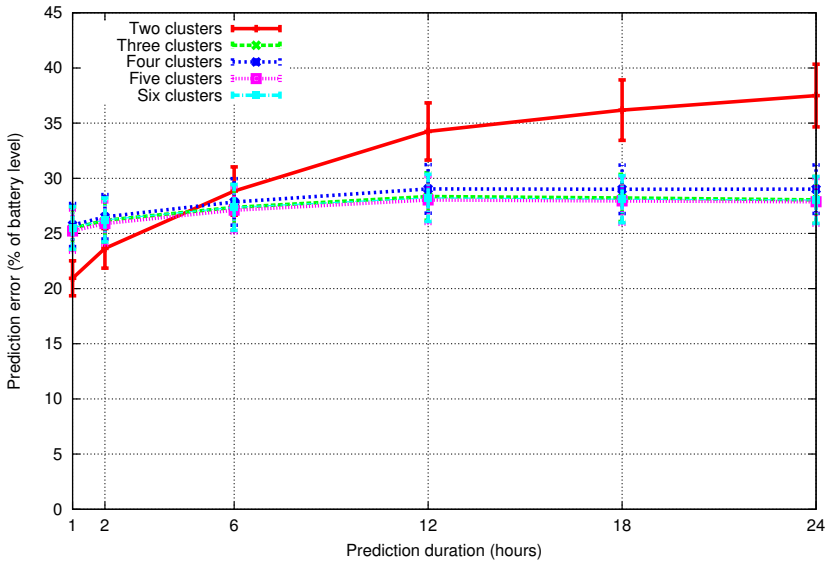


Figure 7.5: Prediction error by cluster size. Clusters of size 3 and above yield statistically equivalent prediction error.

$t_{pred} = \{1, 2, 6, 12, 18, 24\}$ hours using three-way cross validation. We calculate prediction error by iterating through each user's energy trace file in five-minute steps and apply the algorithm using the device's current state. The prediction error for each prediction is taken as the absolute difference between the predicted battery level and the true battery level in the underlying dataset. The mean prediction error over all trials and users is averaged over each cross validation stage to produce the mean prediction error for a given subset of characteristics and cluster count (k).

At the end of this iterative process we found that clustering users by their mean weekday and weekend charge and discharge durations yielded the highest prediction accuracy. Prediction accuracy was the greatest for three clusters as illustrated in Figure 7.5. We call these clusters as opportunistic chargers, light consumers, and nighttime chargers.

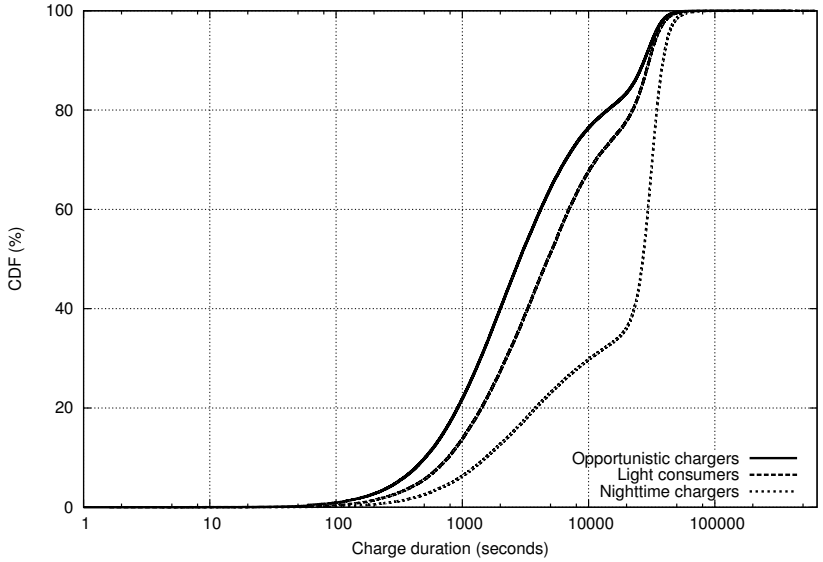


Figure 7.6: CDF of smartphone charge duration by user-type.

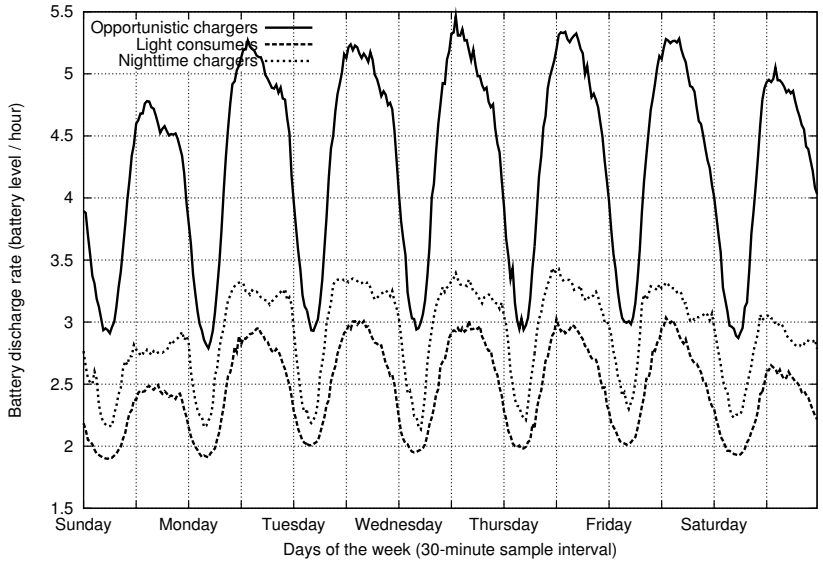


Figure 7.7: Smartphone discharge rate over the week.

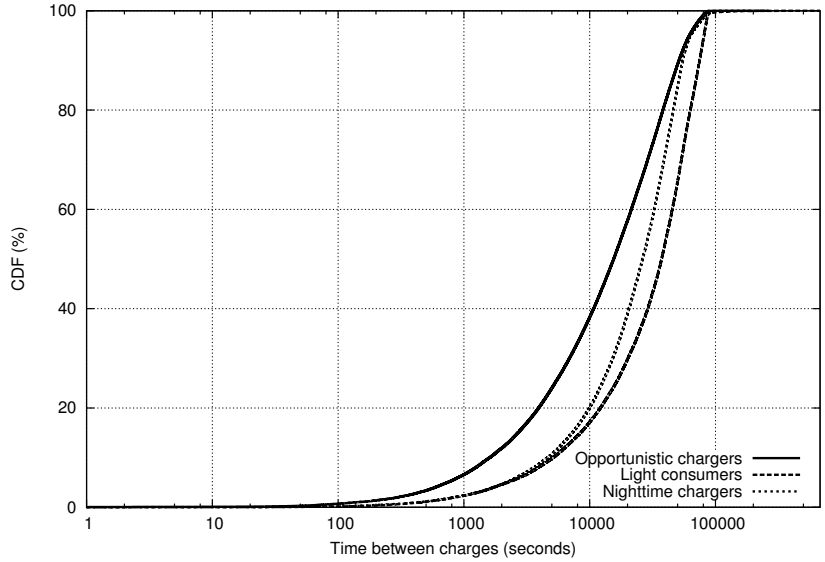


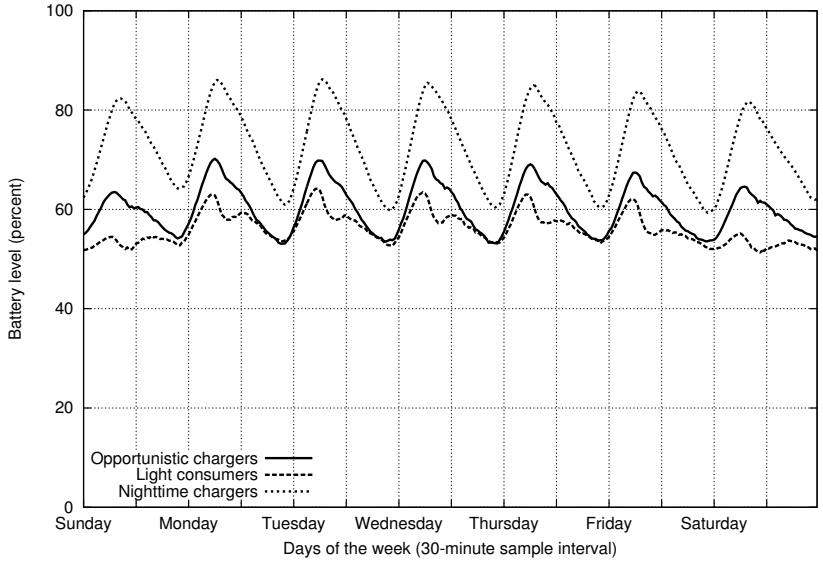
Figure 7.8: CDF of smartphone discharge duration by user-type.

7.5.3 User classification

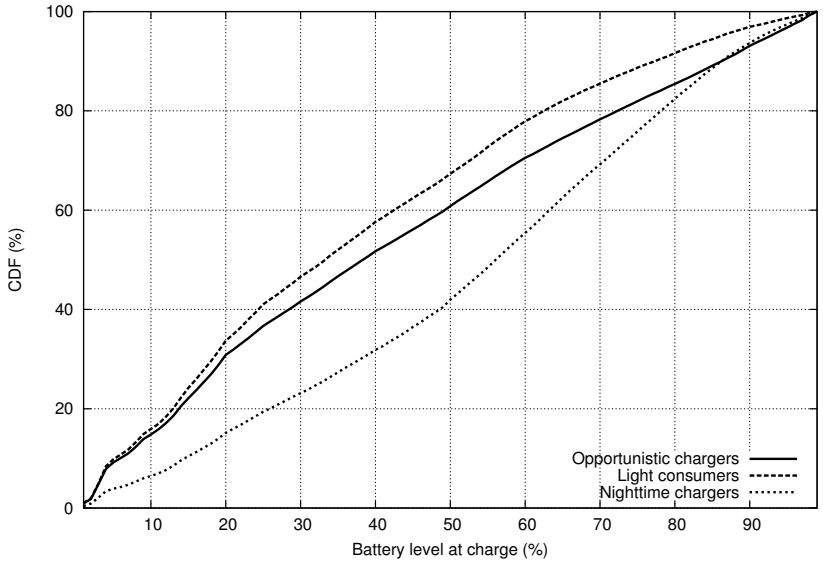
Before a formal evaluation of the predictor, we briefly describe each user-type and their differentiating characteristics.

Opportunistic chargers: Opportunistic chargers are the most common type of smartphone users and represent approximately 63% of the population. These users are primarily characterised by frequent, short charge durations during the hours of 8am to 5pm. The CDF of charge duration for opportunistic chargers is illustrated in Figure 7.6. Of the three user-types, these users are the most aggressive energy consumers, consuming nearly 4.8% of their device's energy per hour. The discharge rates for each user-type throughout the week are illustrated in Figure 7.7.

Light consumers: Light consumers have the lowest energy discharge rate among all three user-types. These users represent approximately 20% of the population. These users charge for longer durations than opportunistic chargers, but discharge their devices over a longer duration. The CDF of discharge duration is illustrated in Figure 7.8. Despite having the lowest discharge rate, light consumers surprisingly maintain the lowest mean



(a) Mean smartphone battery level over the week.



(b) CDF of smartphone battery level when initiating a charge.

Figure 7.9: Smartphone battery statistics.

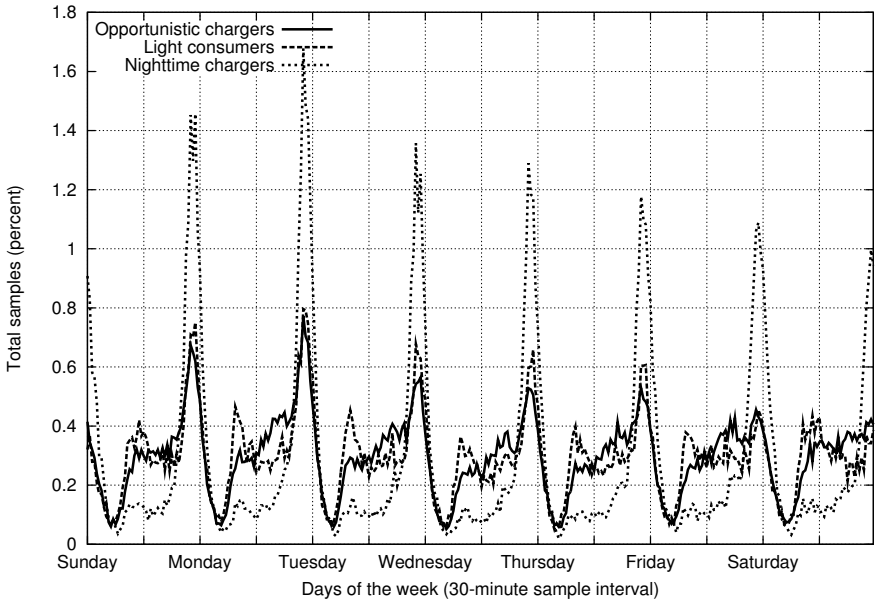
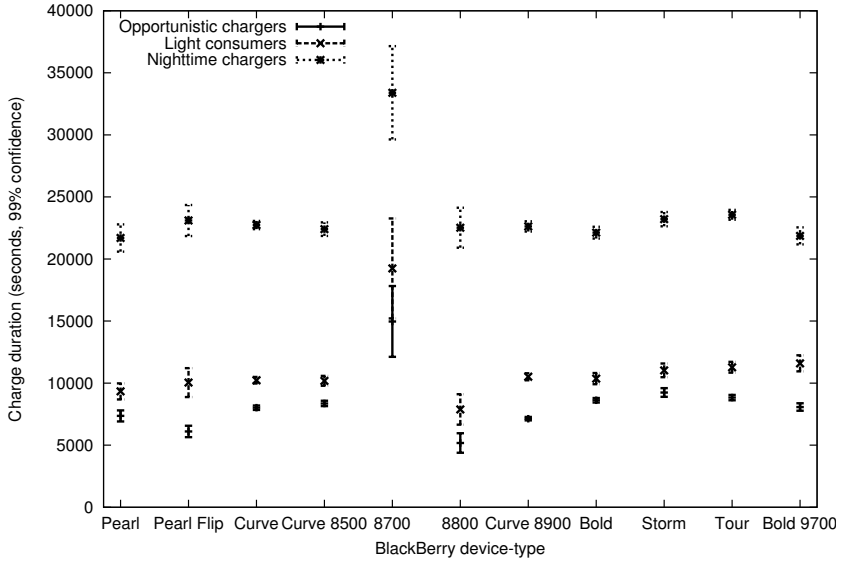


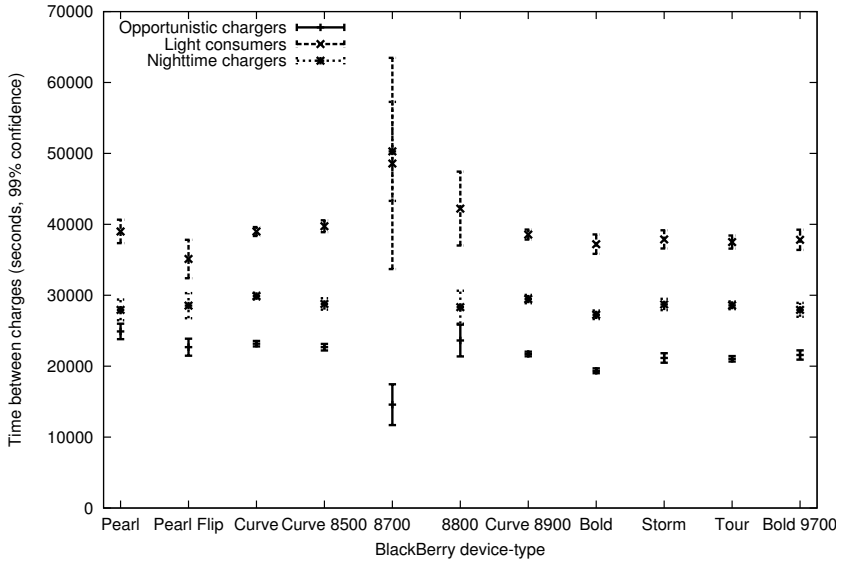
Figure 7.10: PDF of smartphone charge initiation time.

battery level of 56.0% as illustrated in Figure 7.9(a). These users also allow their battery to drop to its lowest level before initiating a charge. On average, a light consumer initiates a charge when their battery level has dropped to 34%. The CDF of battery level when a charge is initiated is illustrated in Figure 7.9(b).

Nighttime chargers: Our final class of user is those that charge at night. These users represent 17% of the population. The charging behaviour of the user is best illustrated in Figure 7.10 as the PDF of the time that users initiate a charge. The daily spike between the hours of 10pm to 11pm illustrate that these users initiate a charge (probably) before going to bed. Given that these users charge predominantly during the night, their mean charge duration is significantly higher than the other two groups, as illustrated in Figure 7.6. Although nighttime chargers consume only 0.5% more of their battery per hour than light consumers, their long charging durations serve to maintain a mean battery level of 72.5%. Similarly, nighttime chargers initiate a charge at an average battery level of 56%.



(a) Charge duration for each user-type and device-type.



(b) Discharge duration for each user-type and device-type.

Figure 7.11: Durations by user and device-type.

Device independence

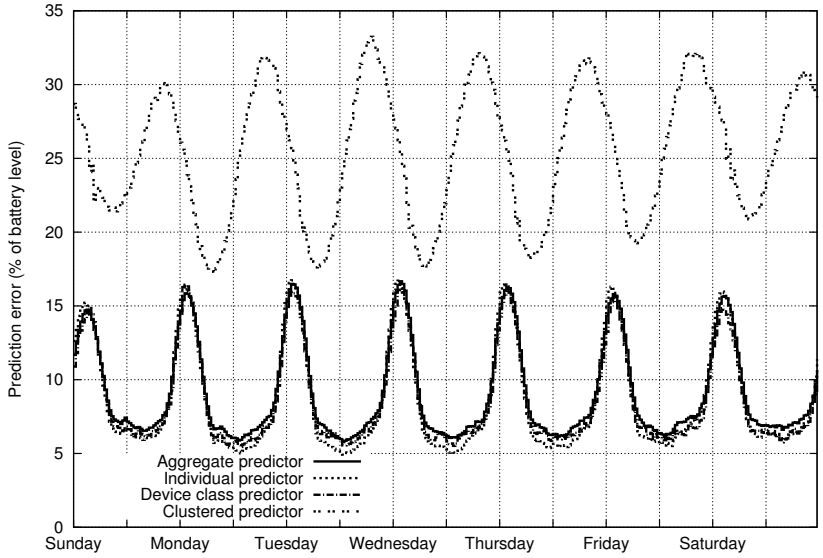
We briefly address the impact of device hardware on our two clustering parameters: charge and discharge duration. Using a t-test, we found that there was no significant difference between device-types at a 99% CI. These results are illustrated in Figure 7.11(a). With the exception of the 8700, nearly every 99% CI contains the each cluster's population mean. We observe similar results in the discharge case, as illustrated in Figure 7.11(b). Although there is more variation among the CIs for each device, we observe that nearly every CI contains the cluster's population mean. The user's charge and discharge characteristics are clearly independent of the device-type used.

7.5.4 Predictor analysis

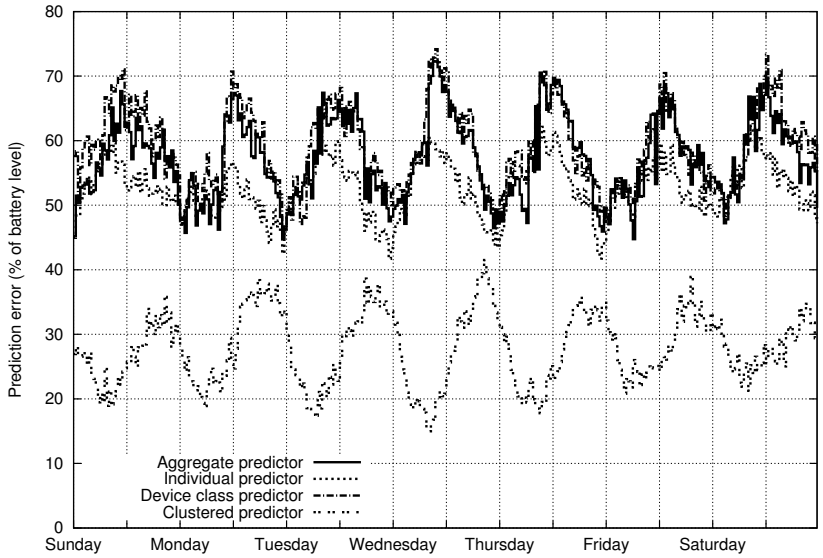
We evaluate the accuracy of user classification-based prediction (*Clustered Predictor*) by comparing it to predictions based on the entire participant population (*Aggregate Predictor*), device-type (*Device Predictor*), and predictions made on an individual basis (*Individual Predictor*). Using the baseline prediction algorithm, we alternate between predictors by initializing the algorithm with the δ and ρ statistics of the training population. The initialization process for each predictor is described below. Although there are many possible predictors, we believe that this set demonstrates the benefit from a user classification-based prediction scheme.

We evaluate the Aggregate, Device, and Clustered Predictors through three-way cross validation of our dataset. We train each predictor on two thirds of the population, the *training set*, and apply the predictor to the remaining third of the dataset, the *target set*. For the Individual Predictor, we train the predictor on each participant's trace then evaluate it against the same trace. Applying the predictor on a participant's traces requires that we iterate through all energy trace files in five-minute steps. At each step through the trace log, we apply each predictor, store the future predicted battery level, and compare the current true value with a previous prediction. We store the absolute error for each prediction for the current bucket. This process is repeated for two remaining portions of the dataset. After applying the predictor to each third of the dataset, we compute the mean absolute error for each bucket in the week.

For the Aggregate Predictor, the population consists of the entire dataset. For the



(a) One hour mean absolute prediction error.



(b) One day absolute mean absolute prediction error.

Figure 7.12: Mean absolute prediction error.

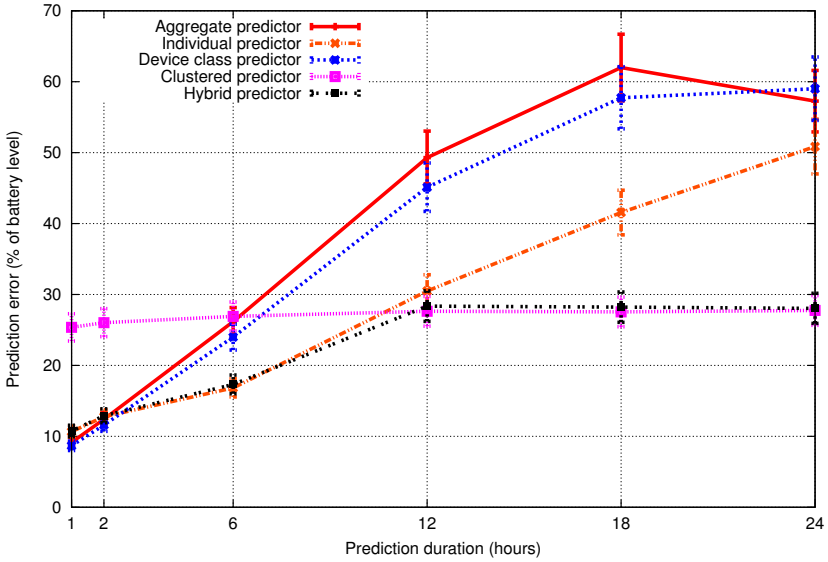


Figure 7.13: Mean absolute prediction error.

Device Predictor, we first partition the dataset into the training and target sets. Each set is subsequently partitioned by device-type. The predictor is then executed using the training set and target set corresponding to each device-type. Similarly, we evaluate the Clustered Predictor by clustering users within the training set and target set into three clusters per set using the same characteristics used to derive our three user-types. For each of the three user-types, we train the predictor on the cluster in the training set and apply it to the corresponding cluster in the target set. As before, the absolute error for each battery level prediction algorithm is stored in a bucket, and the mean absolute error is computed over the course of the week.

Figure 7.12(a) and Figure 7.12(b) illustrate the mean absolute prediction error for one and 24 hour predictions for each of the four prediction algorithms. We illustrate the mean prediction error up to 24 hours in advance from all five predictors in Figure 7.13. These figures illustrate several important findings. Over short durations, predictions made on an individual basis, on an aggregate basis, and by device are significantly more accurate than those based on the clustered user-type. The relationship between the Individual and Clustered Predictors was not expected. At predictions of approximately 12 hours, the

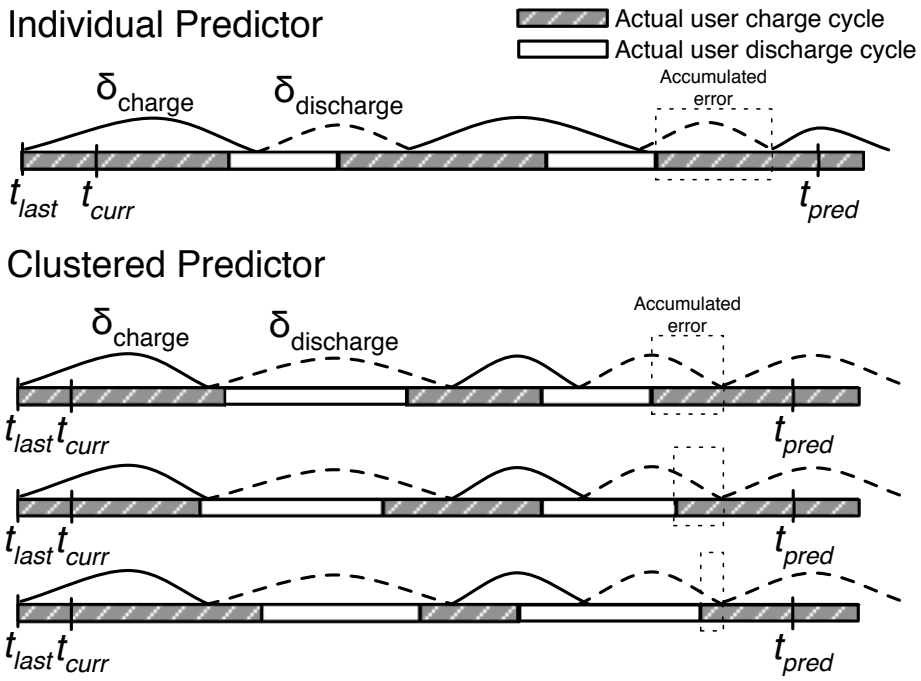


Figure 7.14: Example of error accumulation using four hypothetical traces.

two predictors yield equivalent results. For predications 12 hours and longer, predictions based on aggregate statistics and device-type perform the worst. This is not surprising given that we have shown that users differ, a property that is ignored by the Aggregate Predictor. As previously illustrated in Figure 7.11, there is also more variability in the cluster parameters between user-types than between users of the same device-type. It is therefore not surprising that the Device Predictor also performs poorly.

Over long durations, the Clustered Predictor outperforms the Individual Predictor due to reduced error accumulation. In the Individual Predictor case, each element of the δ vector is calculated over few samples and has a correspondingly high standard deviation. In the Clustered Predictor case, each element of the δ vector is calculated as the mean of the corresponding element of δ for each constituent user (a mean of a mean), which is a closer approximation of the population mean, with a reduced standard deviation and lower propensity to accumulate error. We illustrate this explanation using four hypothetical traces in Figure 7.14. In this example, the Individual and Clustered Predictors are

initialized with δ values derived from either the individual user's trace or from a cluster of users. Each hypothetical trace contain the actual charge and discharge durations for the hypothetical user. Given the time of a prediction, t_{curr}, the time of the last charge or discharge event, t_{last}, and the prediction duration, t_{pred}, the example illustrates how expected charge and discharge durations may overshoot or undershoot the true charge and discharge cycles. The degree that a predicted charge and discharge cycle differs from the true charge and discharge cycle is illustrated as the accumulated error.

The peaks in mean error in the evenings of both Figure 7.12(a) and Figure 7.12(b) are attributed to the order of magnitude in disparity between the charge and discharge rates. Recall from Figure 7.10 that all three user-types tend to charge their device at night. Also recall that our baseline prediction algorithm operates by estimating the duration of a charge or discharge cycle. An incorrect estimation of either duration results in a rapid gap between the predicted and true battery level.

Drawing from these observations, we introduce a fifth predictor, the *Hybrid Predictor*, that combines the short-term accuracy of the Individual Predictor with the long-term accuracy of the Clustered Predictor. The Hybrid Predictor is initialized with both an individual user's energy characteristics, the characteristics of all three user-types, and the centroid coordinate of each user-type's cluster. Recall that users are clustered based on their mean charge and discharge durations for both the weekday and weekend. The centroid is therefore the 4-dimensional value that lies at the center of each user-type cluster. For predictions of less than 12 hours, the Hybrid Predictor yields the same results as the Individual Predictor. For predictions greater than or equal to 12 hours, the Hybrid Predictor examines the individual user's δ values and selects the user's type based on the closest cluster centroid. The predictor then exploits the statistics underlying the user-type to provide predictions as if they were from the Clustered Predictor. In practice, both the energy characteristics of the three user-types and their centroid coordinates would be hard coded into the predictor; allowing the code to be embedded into an energy-intensive application to actively predict the device's future energy level.

7.5.5 Energy Management Oracle

Using the Hybrid Predictor, and the ability to predict the future battery level, we now present the design and analysis of the EMO library. The EMO is designed to provide

energy-aware hints to applications at runtime. Using the library, applications query the EMO prior to initiating an energy intensive operation. Upon receiving a query, the EMO samples the current battery level and decrements the value by the amount of energy that would be consumed by the operation if it were to be executed. The EMO then initiates a 24 hour battery level prediction using the reduced battery level value as the starting point for prediction. The EMO query returns 'true' if the application can safely execute the operation and thus reduce the battery level to a safe level. The EMO otherwise returns 'false' to indicate that the operation will reduce the battery level to a state that will result in the full depletion of the battery.

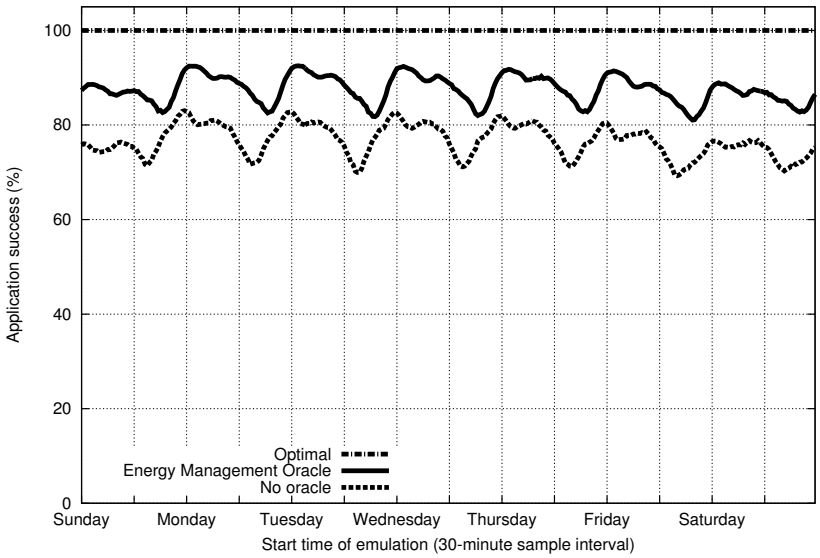
Implementation

Our implementation of the EMO modifies the Hybrid Predictor slightly by halting the prediction if the predicted battery level drops to zero. Like the EET, the EMO is implemented as a stand-alone PERL module that can be imported into any PERL program/script. PERL is supported on a wide range of mobile devices. For devices such as the BlackBerry and iPhone that do not support PERL, the algorithms underlying the EMO could be implemented in Java or Objective C easily.

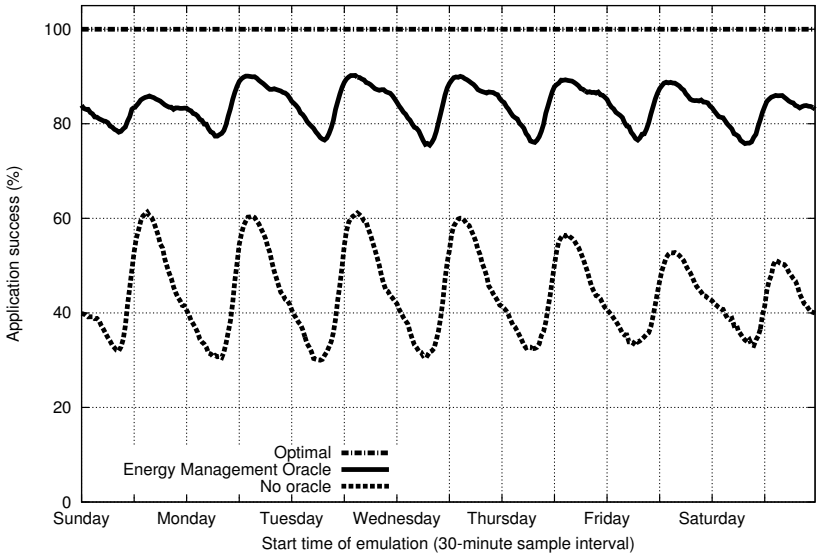
Evaluation

We evaluate the EMO through simulation of the sample application previously described. Recall that the application has two energy intensive operations: uploading 100 MB and downloading 100 MB to a file. We refer to these jointly as the *optional operation*, which consumes approximately 1727 Joules of energy when executed. If the optional operation is performed, the application will consume energy at a *standard rate* of approximately 322 J/hour by scanning neighbouring devices and uploading memory resident data.

Our simulation engine consists of a modified version of the EET that decrements the battery level at the standard rate over a specified execution duration. Like the EET, the simulator steps through each energy trace file in five-minute steps. In each step, it initiates a simulated execution of the application. During each simulated hour, the application queries the EMO to decide if it should perform the optional operation. The simulator decrements the battery level by 1727 Joules if and only if the EMO returns 'true'. Like



(a) Six-hour execution duration using the EMO.



(b) Twelve-hour execution duration using the EMO.

Figure 7.15: Evaluation of the EMO.

the EET, the simulator records a success or failure for each bucket, which allows us to compute the successful execution rate as a function of the starting time of the application.

We compare the results of the simulation with two additional scenarios that demonstrate the upper and lower bounds on the execution success rate. The first, optimal scenario uses an oracle with perfect future knowledge; this oracle scans ahead in the trace file and returns 'true' if and only if the application will succeed. If it returns 'false' the optional operation is not carried out. The lower bound is determined by running the EET as normal, where the simulated application executes without the benefit of battery prediction and the EMO. Figure 7.15(a) and Figure 7.15(b) illustrates the aggregate successful execution rate of the sample application over a six and twelve hour period respectively. Over six hours, the EMO is able to increase the successful execution rate from 76.7% to 87.6%; a 46.5% reduction in failures. Over long durations, the gain is more substantial. Without the EMO, the application exhibits a successful execution rate of only 43.8%. Applying the EMO increases this rate to 84.6%, which amounts to a 93.1% improvement. The EMO does not improve the execution rate to 100% because of prediction errors. We therefore believe that the EMO can add significant value to energy intensive applications.

7.6 Chapter summary

This chapter has detailed the design and deployment of a large-scale smartphone user study that examines how users interact with and consume energy on their personal mobile devices. Our dataset contains over 1150 years of cumulative data for 20,100 users from around the globe. Smartphone users, like all complex objects, can be classified by many variables. Our work differentiates users on the basis of their daily battery charge and discharge characteristics, and has identified three user-types: opportunistic chargers, light consumers, and nighttime chargers. This classification scheme provides 72% accuracy on predictions between 12 to 24 hours in advance. Using our predictor we built the Energy Management Oracle library, which can be queried by applications prior to the execution of an energy intensive operation. Using the oracle, we demonstrate that applications can achieve a near optimal successful execution rate. Using our dataset, we expect that other researchers will discover more efficient energy prediction techniques and that user differentiation will continue to yield similar improvements in prediction accuracy.

As the scale and complexity of mobile applications continue to increase, we believe that energy consumption will become a new dimension in mobile software evaluation. Applications that consume large amounts of energy over long durations will be practical for only a subset of the user population with compatible energy characteristics. We predict that in the future, software will be distributed with a specified energy demand in the same way that resources such as CPU speed, memory/disk capacity, and network bandwidth requirements are specified in today's software. The Energy Emulation Toolkit is designed to support this trend by providing developers with tools to evaluate their applications across all users, device hardware, and user-types.

Chapter 8

Conclusion and Future Work

8.1 Summary

Censorship is currently a fact of life for many people in the world. The recent efforts to quell political uprisings throughout the Middle East are evidence for the need for censorship-resistant communication systems [154]. CTNs running on personal mobile devices are intended to counteract this growing trend.

Building on lessons learned from previous DTN systems, this thesis has defined an architecture for CTNs. In a CTN, censorship resistance is achieved through nodes forming pairwise trust relationships with other mutually trusted nodes. Trust relationships are bootstrapped by direct social interaction between pairs of nodes. In a CTN, the underlying trust topology serves to constrain both the degree of information shared with other nodes and information that must be maintained about other nodes. The formation of trust relationships and the ensuing exchange of identities further facilitate private communication between mutually trusted nodes. The trust topology is further utilized to provide a scope for which to disseminate metadata about content in the system. By maintaining metadata corresponding to reachable content, our approach tightly couples content in the system with nodes' demand for it. This approach allows nodes to explicitly and unambiguously express their demand for content in the network.

We define the Laissez-faire framework, an incentivized, free-rider tolerant approach to routing content within a DTN. Unlike existing approaches to DTN routing, content is pulled from its source to destination to satisfy inter-node debt obligations and to satisfy content requests. Unlike existing DTN routing protocols, Laissez-faire does not mandate any specific mechanism for valuing and selecting fragments. Nodes implementing Laissez-faire are free to exchange content in anyway they choose; however, nodes that do not exchange content with others will accumulate debt and eventually be starved from the system by resource-sensitive nodes. We believe that our simple approach to maintaining the debt relationship between nodes makes our framework robust to free-riding nodes that can manipulate any routing protocol for their own benefit. We evaluate several example strategies for ranking nodes and several strategies for selecting fragments from a neighbouring node. Further, we believe DTN research that is designed for resource-sensitive smartphone users can benefit by our selfish approach to node interaction.

Through a prototype BlackBerry implementation of our CTN system, we demonstrate the feasibility of CTNs in both controlled experiments and a three-week field trial of 36

devices at the University of Waterloo. Although our field trial assumes that all nodes are mutually trusted and is naturally biased by the mobility and diurnal patterns of graduate students, our field trial was successful in exchanging large quantities of content when all nodes both create content and express demand for each other's content. Our field trial further highlights the value of the occasional use of a low-bandwidth, SMS-based control channel.

We conduct the first work that measures the channel characteristics of 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 block SMS bursts, 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. We believe that our work could serve as a basis for future study of the channel characteristics of SMS - especially considering that SMS use continues to grow [78].

Building on our understanding of channel characteristics of SMS, we design an SMS-based transport protocol for use as a dedicated CTN control channel. Through a Java implementation, we showed 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 to exchange data over SMS. This work is useful when alternatives to SMS are either expensive, poorly deployed, or do not exist.

CTNs are a class of application that must run continuously to maximize connection opportunities, and autonomously since the user should not be required to manage the CTN. CTNs also consume a larger quantity of energy compared to most smartphone applications [52]. We conducted a large-scale smartphone user study that examines how users interact with and consume energy on their personal mobile devices. As the scale and complexity of mobile applications continue to increase, we believe that energy consumption will become a new dimension in mobile software evaluation. We built the Energy Emulation Toolkit to support this trend by providing developers with tools to evaluate their applications across all users, BlackBerry device hardware, and user-types.

We demonstrated that smartphone users can be classified based upon their daily battery charge and discharge characteristics, and have identified three user-types: opportunistic

chargers, light consumers, and nighttime chargers. This classification scheme provides 72% accuracy on predictions between 12 to 24 hours in advance. Using our predictor we built the Energy Management Oracle library, which can be queried by applications prior to the execution of an energy intensive operation.

8.1.1 Summary of achieved design goals and constraints

The following sections describe how each of the four censorship resistance criteria and the six design constraints outlined in Chapter 1 have been satisfied. We further discuss how the limitations of our prototype implementation affect each property.

Robustness

Recall from Reference [155] that robustness, in the context of CTNs, requires that no third party is able to prevent two CTN nodes from communicating with each other. Through opportunistic, short-range wireless communication, CTN nodes may exchange content directly and trivially achieve robustness. Attackers of our system are assumed to have the capability of blocking communication in isolated areas. Robustness may, therefore, be temporarily suspended while CTN nodes occupy regions blocked by an attacker. However, an attacker is not capable of blocking communication everywhere, and thus cannot prevent two nodes from communicating indefinitely.

Our prototype implementation clearly does not satisfy this requirement. An attack that disables the BLS, for example, could easily deny communication to all nodes. We discuss how our prototype will evolve to be fully independent of infrastructure, and thus satisfy the robustness criteria, later in this section.

Communication over the SMS control channel can easily be blocked, further violating robustness of our system. However, as discussed in Chapter 6, the user of SMS as a control channel is optional. In the absence of SMS, control messages can be delivered opportunistically along with content fragments. We view our work as providing a tradeoff between communication robustness and delay that can be tailored by users to meet their specific circumstances.

Scalability

All nodes in a CTN operate by communicating and sharing content only with nodes whom they trust. Consequently, as new nodes enter a CTN and the communication graph underlying the network (the trust topology) grows, node interaction remains bounded to a limited set. The storage space requirement for nodes participating in a CTN is bounded by the number of nodes they trust times the quantity of data (the debt threshold) that a node is willing to allocate to others.

As the system scales and the underlying trust graph grows, the quantity of metadata stored within the Content Space grows exponentially. However, our prototype implementation naïvely pruned the Content Space based on a least-recently encountered policy once the Content Space reached a specified size; however, given the importance of the Content Space in our design, further study of pruning the Content Space is required. We discuss this area of future work in Section 8.2.1.

Privacy

Communication privacy requires that no third party is able to determine the contents of a messages exchanged between two CTN nodes. Each node in a CTN maintains a private/public key pair. The public key is shared with trusted nodes as part of a node's identity. During an opportunistic connection between two nodes, each set of keys are used to exchange a session key. The session key is used to symmetrically encrypt communication between a pair of nodes. Using off-the-shelf, commodity encryption libraries, our prototype implementation easily achieves communication privacy.

We explore other aspects of privacy in the next two sections.

Strong unlinkability

Strong unlinkability requires that no third party is able to determine that two CTN nodes are communicating. Unfortunately, given the assumption in our threat model that an attacker may monitor specific regions and subsequently identify communication patterns, this constraint cannot be guaranteed 100% of the time. It is possible for an attacker, in for example, a public square, to identity devices that are communicating with each other.

Although the content exchanged between nodes is obfuscated through encryption, preventing an attacker from identifying the nature of the communication, repeated communication between nodes may be evidence of illicit activity. One approach to solving this problem is steganography - embedding the encrypted content within other transmissions [14, 82]. For example, two devices that want to exchange content may simply stream non-censored content to another node. When transmitting the non-censored content, the transmitting node may introduce noise into the transmitted signal that encodes the censored content fragment [20]. Although this technique does not provide unlinkability, it provides plausible deniability in the event that two nodes are identified as communicating with each other.

The use of SMS in a CTN violates the unlinkability constraint due to its dependency on cellular network infrastructure. As discussed above, the use of SMS as a control channel is optional. Nodes that do not wish to use SMS simply omit their phone number when exchanging an identity with another node. Control messages would then be delivered directly exclusively through opportunistic connections.

Identity privacy

Content within a CTN is represented by metadata, which contains the GUID(s) of all owners of a content item. Recall that a GUID is a self assigned random number that uniquely identifies a node, and is shared as part of a node's identity. Content metadata is only shared with mutually trusted nodes; however, metadata may propagate beyond the set of nodes trusted by the content owner. Thus allowing untrusted nodes to discover the content owned by the GUID.

Our approach to establishing trust within a CTN relies on existing social relationships. Nodes only trust each other in a CTN if their users share trust in the real world. We therefore assume that a node will never betray the trust of another node by disclosing its identity to any other node in the system. Given this assumption, it is not possible for an untrusted node to associate a GUID of a content item with the identity of its owner.

Infrastructure-independence

Our CTN architecture and Laissez-faire framework do not rely on infrastructure to operate. However, our prototype BlackBerry implementation is infrastructure dependent due to the

lack of support for Wi-Fi ad hoc mode on BlackBerry devices as of April 2012 [6]. To overcome this hardware limitations, our prototype must operate in Wi-Fi infrastructure mode. After having discovered a neighbouring node through an infrastructure-independent Bluetooth scan, nodes connect to each other over Wi-Fi. To connect to a neighbouring node over Wi-Fi requires its IP address. Until recently, BlackBerry devices were unable to receive broadcast UDP packets, leaving no mechanism to discover the dynamic IP addresses of older, neighbouring devices. The BLS was introduced to overcome this additional hardware limitation. The BLS is an online service that maps a Bluetooth MAC addresses to the LAN and WAN IP addresses of neighbouring devices. The use of the BLS further violates our infrastructure-independence goal; however, we expect that future versions of the system will utilize Wi-Fi Direct, which is available as of version 4.0 of the Android platform [131].

Openness

Our CTN architecture achieves openness by virtue of being open source software that may be downloaded and modified by anyone in the world. In Chapter 2, we demonstrated that prior DTN work is ill-equipped to operate under this requirement. Existing work, if made open source and widely available, could easily be modified to create free-riding nodes and game the system. As an increasing number of nodes become free-riders, it is easy to conceive scenarios where little content is propagated through a network of resource-sensitive nodes. Although our CTN architecture is not immune to free-riding, the tit-for-tat accounting mechanisms that underlines the Laissez-faire framework can be used to restrict the amount of content provided to free-riding nodes.

No specified forwarding protocol

A CTN is able to be distributed as open source software because it does not mandate that nodes implement any specific routing protocol. Nodes in a CTN are only required to implement the tit-for-tat protocol underlying the Laissez-faire framework. Under Laissez-faire, all nodes are free to operate selfishly to satisfy their own demands for content. A node cannot cannot mandate that another node behave it a specific way. It may only adapt its behaviour to incentivize behaviour on other nodes. For example, to incentivize another node to retrieve content, a node can proactively retrieve content on their behalf. Credit is

then accumulated when the content is delivered. Although the receiving node is still not required to retrieve any content, they will eventually need to retrieve content to satisfy their debt relationship with the other node. We believe that this selfish approach to node interaction will be an attribute that similar systems will adopt to achieve openness.

Deployability

As open source software, our prototype CTN system may be downloaded, compiled, and installed by anyone in the world. In practice, we believe that content gateways, introduced in Section 1.2.2, will be instrumental in the initial deployment of CTNs. Users may install the CTN software without the need to register with an central server. Moreover, identities in the system are self-generated by each node and may be verified by other nodes using standard PKI techniques.

Participation fluidity

As previously discussed, the CTN software is open source software and available to all and anyone may bootstrap their own CTN (of size 1), and incrementally add trusted nodes to expand the size of the network. As nodes have no formal responsibility to other nodes in the system, nodes may leave the network at any time by either deleting the CTN software or by ceasing to trust other nodes. Nodes that leave the system leave behind any debt or credit accumulated with other nodes. We assume that since nodes only communicate with other nodes whose users share a real-world trust relationship, this process would not be taken advantage of in practice. Nodes that repeatedly create new identities are clearly gaming the system.

Nodes also leave the system with the identities of nodes that are trusted. Although the identities are the primary asset of interest to attackers, we do not consider this a threat to achieving participation fluidity because of the real-world trust relationships that underpin the exchange of identities. We believe that individuals that trust each other in the real world would not betray each other after leaving a CTN.

Attacking nodes that successfully infiltrate a CTN, and subsequently leave the network, are not guaranteed to protect the identities of other nodes. We discuss this exception in Section 8.1.3.

8.1.2 Design limitations

The accumulation of credit and debt that form the debt relationship between nodes is fundamental to the operation of CTNs. However, this property can lead to a design limitation. For example, in the field trial performed in Chapter 5, all nodes demanded content on an hourly basis. The result was a highly symmetric relationship in quantity of data demanded by each node. This observation is illustrated by the graphs in Appendix B that indicate the quantity of content demanded by each node over the course of field trial. Throughout the field trial the debt threshold never needed to be enforced. However, since all nodes were resource-sensitive, had demand been asymmetric, we would expect many nodes to have reached their debt threshold and denied other nodes' requests for content fragments. Although this would have been an intentional side-effect of nodes' resistance to free-riding, it is a natural state that can adversely affect the performance of the system. It is possible for a network consisting purely of resource-sensitive nodes to perform poorly if demand is asymmetric.

We address two approaches to mitigating the affect of this limitation in Section 8.2.3 and Section 8.2.4.

8.1.3 Security analysis

In Chapter 1 we outlined our threat model and the capabilities of a attacker. The most capable attacker of our system is the state. Determining the identity of individuals that violate state-mandated censorship is assumed to be the main asset of interest to attackers.

In this section we review each capability and assess how our system protects, or fails to protect, the identity of CTN users.

Internet-monitoring

Clearly, a prototype implementation that uses the BLS and SMS cannot resist attacks by an attacker with the capability to monitor the Internet. However, as previously discussed, the use of SMS as a control channel in optional and disabled by default. The BLS is only used to overcome hardware limitations of our testbed.

The CTN architecture and Laissez-faire framework defined in this thesis are designed to operate without using Internet infrastructure. By not using Internet infrastructure, our system is robust to Internet-monitoring based attacks.

Locate end-hosts

As a corollary to not depending on Internet infrastructure, and thus being robust to Internet-based attacks, our system is robust to an attacker's ability to locate Internet end-hosts.

Regulate access to public Internet access points

Our prototype CTN application is currently dependent on WLAN infrastructure due to the lack of support for Wi-Fi ad hoc mode on BlackBerry devices. However, future implementations will utilize Wi-Fi direct and will therefore not utilize access points. By constraining communication between nodes to direct, wireless, ad hoc communication, CTNs will be resistant to government regulation of WLAN access points.

Isolated monitoring

Our system encrypts all communication between nodes, and thus prevents an attacker from identifying the nature of the communication. However, utilizing wireless, ad hoc communication may be sufficient evidence that a node is participating in the exchange of censored content. Since attackers are assumed to have the ability to monitor and identify communication patterns in an isolated region, nodes in our system are vulnerable to detection and subsequent identification.

As discussed above, steganography is a technique that could be used to obfuscate communication between a pair of nodes. By embedding the exchange of censored content within, for example, a stream of non-censored music, two nodes could safely exchange content over a wireless, opportunistic connection with a plausible explanation.

Infiltrate social networks

In a real-world deployment, it is inevitable that agents of the government would install the CTN software and seek to determine the content being exchanged within the network. We assume that attackers have the ability to gain the trust of some nodes, and thus obtain information about the content in the network. The CTN architecture takes steps to mitigate the degree of this attack.

In this scenario, the node that inadvertently establishes trust with the attacking node is totally vulnerable. Their identity would be disclosed to the attacker as part of the establishment of a trust relationship. The metadata of content owned or carried by the node would also be disclosed to the attacker, which may immediately incriminate the user. While the compromised node will continue to share metadata corresponding to the rest of the content in the network, the identities of other nodes in the network are not contained within the metadata. The owner of a content item, or fragments of a content item, are represented by their GUID only, which is self-assigned, randomly generated number. An attacker may therefore discover the GUIDs of content owners, the number of participants in the network (assuming that each node limits itself to one GUID), but it may not discover the identity of the nodes. There is, unfortunately, one exception, which we describe next.

Physical harm

Unfortunately, our system is not robust to attacks that inflict personal harm on individuals. We assume that when faced with physical harm and/or death, users will divulge all information 'requested' by an attacker. This information includes: the password to decrypt the file system of the device [2, 6] and the real-world identities of nodes that it trusts and has communicated with in the past.

Although devices could be equipped with a 'panic button', configured to wipe the device's memory in the case of emergency [122], we consider this measure insufficient to resist attacks involving physical harm. Even if a device is wiped, a determined, ruthless attacker can extract information directly from the user.

8.2 Future work

As a new approach to fighting censorship, there are many areas of future work. We consider the following problems to have the greatest immediate benefit.

8.2.1 Pruning the Content Space

As trust between two CTN nodes is established, the degree of each node in the underlying trust graph increases by one. Over time, as more trust relationships form, the graph will grow and the quantity of metadata that may be propagated through the network will grow exponentially. From the perspective of providing nodes with information about content within the network, more metadata is always preferred to less. However, a large amount of metadata will result in high communication overhead when exchanging metadata updates with neighbouring devices, and at some point overwhelm the link capacity of two nodes. We therefore require a mechanism to constrain the size of the Content Space to a practical size.

Our prototype implementation currently discards content metadata according to a least-recently encountered policy. However, this approach is naïve because it does not consider the contact frequency of nodes, the amount of content previously exchanged between nodes, or the amount of content that nodes demand of each other. Favouring metadata for content of frequently encountered nodes would reduce the delay in retrieving any content that is subsequently demanded. Content that is reachable via nodes with a high debt relationship may be less accessible than content held by nodes with a lower debt relationship. Similarly, if a node A demands a large amount of content from node B, then node A is incentivized to provide content to node B. Given this relationship, it may be beneficial to node B to preserve metadata for content that is reachable via node A.

The *quality* of the content could also be considered when discarding metadata. Although quality is a qualitative metric, it could be inferred by simply ranking nodes by the quantity of their content that has previously been retrieved. Metadata corresponding to content that is created by quality producers is probably more valuable than content of poor quality producers. The user's viewing habits, or whether retrieved content was deleted or saved, could also be used to further refine the quality metric.

Although there are many factors that affect how content within a CTN is delivered, we believe that the quality of metadata can impact service. It is therefore important for future CTN implementations to consider these factors when discarding metadata.

8.2.2 Proactive state propagation

As previously discussed, the use of SMS violates our CTN design requirements and should not be used in jurisdictions where the penalties for participating in a CTN could be severe [110]. However, CTNs may be used in less strict jurisdictions, such as Europe or North America, where participating in a CTN is not sufficient evidence of illegal activity. CTNs may also be used for non-nefarious reasons despite the availability of legal, more convenient, Internet-based alternatives. Under these conditions, we believe that an SMS-based control channel is ideally suited to increase the efficiency of the network. Specifically, we believe that SMS could be used to propagate changes to both the Content Space and demand vectors.

As new content is created or retrieved by a node, Content Space updates would allow other nodes to discover content earlier. This would be particularly useful in disseminating highly popular, viral content to propagate as fast as possible. Conversely, if a node deletes a content item, then an update would allow nodes to take the following action. If the deletion of the content item caused the content to only be partially present in the network, then all other nodes could save storage capacity by deleting their partial copies of the content item. Similarly, nodes would be prevented from wasting a communication opportunity by retrieving a fragment that would never result in a completed content item. We believe that this simple feature would be a valuable addition to a CTN, and it's impact worth exploring in future work. However, the greatest impact on a CTN is likely to stem from propagating demand vector updates.

Redundant content transfers take place whenever two nodes retrieve the same content fragment on a behalf of a single node. Depending on the trust topology and as a disconnected network, redundant data transfers could be common. We believe these redundancies could be reduced, if not eliminated, by proactively propagating demand vector updates. Demand vector updates may contain the CIDs of a newly demanded content item(s), an item that has been fully retrieved, or an item for which demand has been withdrawn. A demand vector update may also contain the CID and index of a fragment(s) that have

been retrieved from other nodes. These updates would allow other nodes to update their view of the node's state and adapt their behaviour accordingly. By proactively propagating demand vector updates, we believe that redundant transfers could be minimized. However, given that this technique is designed to reduce the number of content fragment replicas in the system, its impact on both delay and the delivery ratio is an open area of study.

8.2.3 Debt forgiveness

As previously discussed, it is possible for asymmetries to form in a CTN that can result in some nodes accumulating large amounts of credit, while others accumulate large amounts of debt. Nodes have no obligation to act on debt relationships, and, in practice, we expect that a large portion of nodes, the content activists, will ignore the debt relationship in favour of propagating content. However, for resource-sensitive nodes that choose to strictly enforce the credit and debt relationship, we believe that it may be in their bests interests to forgive debt over time. That is, we believe that selfish strategies may be sub-optimal, even for resource-sensitive nodes that would otherwise implement a pure tit-for-tat selfish strategy. The accumulation of debt can create livelock in the system. Forgiving debt over time would prevent selfish nodes from creating livelock, cause more data to be exchanged, which may result in selfish node receiving more content than they would under a purely selfish strategy. The rate that selfish nodes should forgive debt vs. the benefit of forgiving debt is an open question that we believe is worth exploring in future evaluations of this work.

Debt forgiveness techniques, however, are not immune to attack. In the BitTorrent system, for example, nodes may be retrieve content with minimal reciprocity by first consuming the resources allocated to them by peers. The nodes subsequently exploit peers' forgiveness of debt to continue to retrieve content with minimal reciprocity [109, 141]. In our current CTN design, nodes could exploit debt forgiveness in a similar manner without providing any reciprocal content.

8.2.4 Currency swapping

The pairwise debt relationship between nodes under the Laissez-faire framework creates a byte-based *currency* between CTN nodes. Under this currency, an accumulated byte of

credit can be redeemed for a byte of content. However, this currency is constrained to each pair of nodes for which a debt relationship exists. In addition to forgiving debt obligations, we believe that exchanging this currency with other nodes could be used to mitigate the effect of asymmetries in the network. For example, in an asymmetric deployment, highly altruistic nodes that accumulate a large amount of credit may transfer their credit to other highly indebted nodes in the network. However, given that trust relationships are pairwise and both the topology of the CTN and identities of other nodes may be unknown, the mechanism for swapping currencies between potentially non-trusted nodes is likely a difficult problem to solve.

8.3 Conclusion

Although there are several limitations and avenues for future work, we have shown that our CTN system satisfies our design goals through an infrastructure-independent, delay tolerant architecture that partitions the population, and constrains communication, to pairs of mutually trusted nodes. Our architecture provides communication privacy through a pairwise encrypted channel, and identity privacy is achieved by decoupling content from the identities of nodes that possess it. Our CTN architecture is incrementally deployable. Participants may enter or leave at any time, with negligible impact on the CTN. Therefore there are no barriers to CTNs forming all over a region, continuously coalescing and fragmenting into new CTNs as participants join and leave the network. Finally, our system fully satisfies our goal of openness. By releasing the work as open source (upon publication of this thesis), anyone in the world may download, modify, and deploy their own CTN. As an open system, wide-spread adoption will inevitably bring free-riders eager to capitalize on the altruism of others. We believe that the simplicity of the Laissez-faire framework protects the needs of resource-sensitive nodes and effectively limits the negative impact that free-riders can have on the system.

Feature limitations of today's devices unfortunately force us to make several implementation compromises that violate the criteria of censorship resistance. However, we are optimistic that time will correct these shortcomings to make our system fully infrastructure-independent.

We are under no illusion that this work will defeat censorship; however, we strongly

believe that the approach outlined in this thesis is the most practical approach to solving the growing problem of information censorship in the world.

APPENDICES

Appendix A

Using the MobiTether Prototype CTN System

This appendix was written for participants of the field trial described in Chapter 5. It provided participants with the information needed to interact with the application. This document is included in this thesis both to illustrate the prototype implementation, and to provide context into the field trial.

The prototype implementation originally organized nodes into trust groups, whose underlying trust topology was a *clique*. Each clique had an administrator that was implicitly trusted by every member of the clique. Trust amongst clique members was transitive about the clique administrator. This change in architecture did not affect the field trial because we controlled for the trust topology. All participants were configured to be mutually trusted.

A.1 Application tutorial

The prototype CTN BlackBerry system is named MobiTether. It is a continuously running background process. The user interface component of the application may be launched by clicking on the MobiTether icon on the BlackBerry home screen.

Once launched there are six components of the main screen: a large field that is initially empty representing the media requested by the user, the neighbourhood button, the media

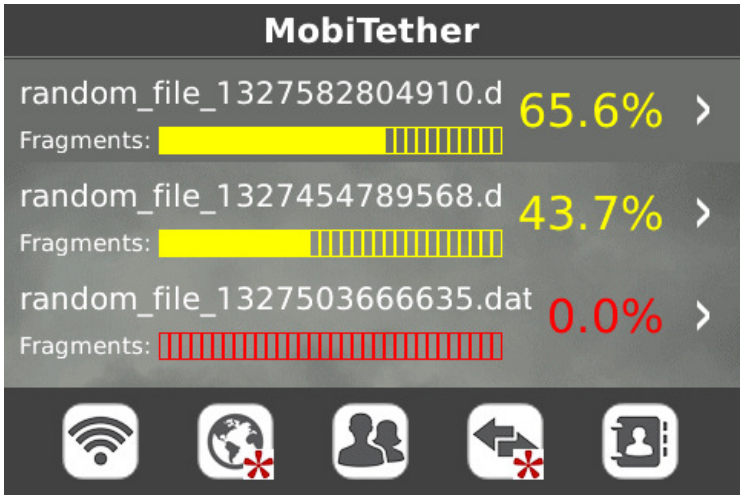


Figure A.1: Main screen of the MobiTether prototype CTN system.

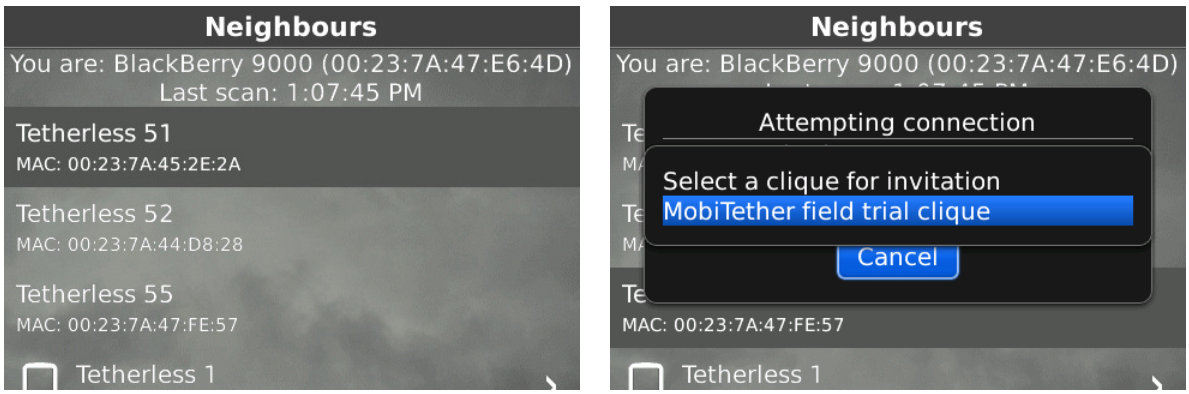
space button, the cliques button, the control message button, and the settings button. The following sub-sections describe each component.

A.1.1 Neighbourhood

The neighbourhood page indicates the set of neighbouring Bluetooth devices. Devices are indicated by their friendly device name (when available) and their Bluetooth MAC address.

By default, the neighbourhood page displays the MAC addresses of all nearby devices. This screen is illustrated in Figure A.2(a). Known, trusted devices may be filtered by selecting `Show trusted devices only` from the menu. Trusted devices are accompanied by a white device icon on the left.

CTNs operate by forming mutually trusted groups of users, known as cliques. Cliques have a single trusted administrator that regulates clique membership. To invite a neighbouring device to a clique, click on the desired neighbour. Clicking on a neighbouring device will initiate a connection attempt. If the connection is successful (i.e. the other device is running the CTN software), then the inviter device will be prompted to select a clique to invite the neighbouring device into. The clique invitation screen is illustrated in Figure A.2(b). This figure illustrates an invitation to `Tetherless 52` into `MobiTether clique 70`.



(a) Select neighbouring Bluetooth enabled device.

(b) Select clique for invitation.

Figure A.2: MobiTether screens for inviting new members to existing cliques.

If the invitee is not the clique administrator, then the administrator's permission must be obtained before adding the invitee to the clique (i.e. disclosing the identities of other clique members). This functionality has been disabled in the field trial. Only the clique administrator may add new members.

A.1.2 Media Space

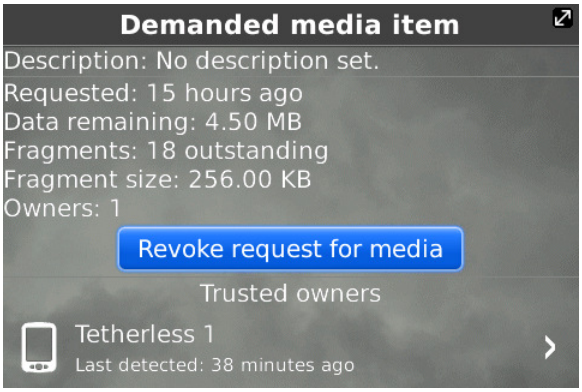
In a CTN, nodes express their demand for content explicitly using a globally unique ID for each known content item. The Media Space[1] displays all content known to the device (both local and remote). Figure A.3(a) illustrates how media is displayed.

Users may express demand for a remote content item (items present on other devices) by selecting the item, which opens a screen that provides additional metadata about the content item. Clicking on the button `Request media` will add the content item to the set of content demanded by the device. Users may similarly revoke their demand for the item by clicking Revoke request for media as illustrated in Figure A.3(b).

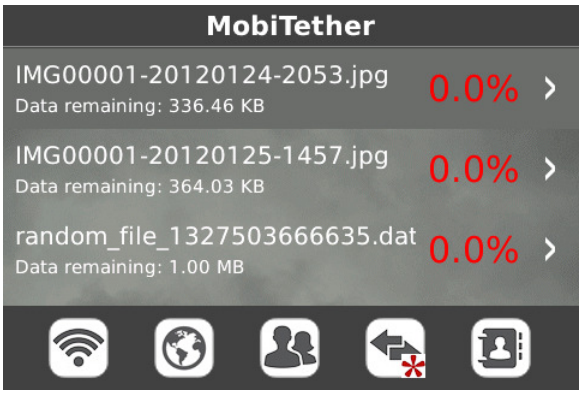
After requesting a content item, the current state of the request is shown on the main screen of the application (Figure A.3(c)). If many items have been requested, the user may sort the list via options in the menu. The requested content item will be visible on the

[1]The Content Space was called the Media Space in the prototype implementation.

(a) Media available on the device (locally).



(b) Revoke request for media.

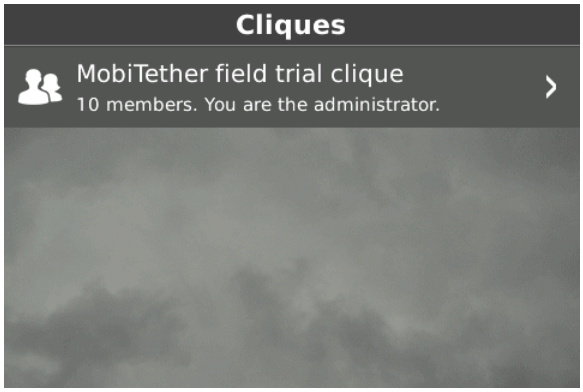


(c) Main screen displays retrieval status of demanded media.

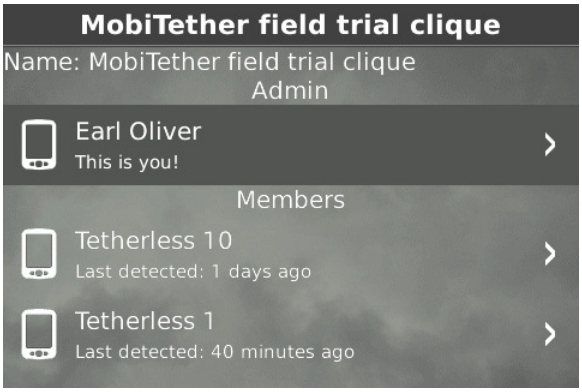
Figure A.3: MobiTether screens for displaying media information.

main screen until it has been fully retrieve, at which point it will be available in the `Local media` list within the Media Space screen.

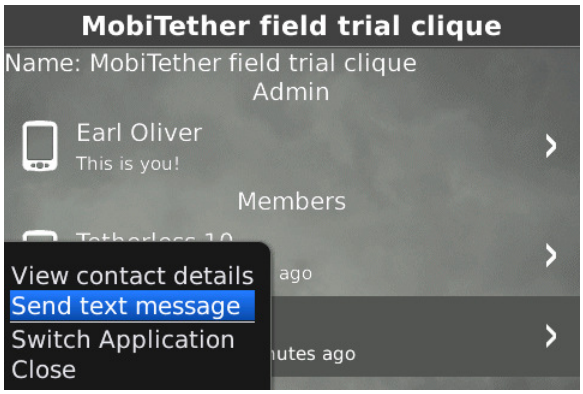
User interaction is not required in this study. Instead, each device has been instrumented to generate a "dummy" media item every hour during the hours of 8am to 10pm. Dummy content items are generated with an artificial popularity index in the range of [0, 9]. Every hour the application will automatically demand a media item according to a power-law distribution. The automatic creation and demand of content is postponed by one hour each time the user creates and/or demands content.



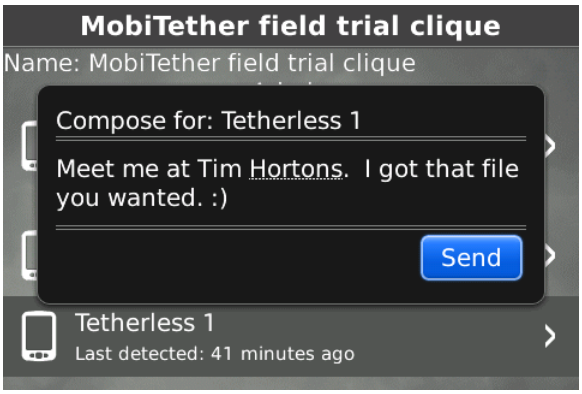
(a) Display a list of constituent cliques.



(b) View clique membership.



(c) Send secure text messages to other clique members.



(d) Coordinate with clique members to reduce delays.

Figure A.4: MobiTether screens for displaying clique state.

A.1.3 Cliques

The Cliques screen illustrates all of the cliques that the local user is a member of (Figure A.4(a)). Through the menu, the user may create new cliques. The ability to leave a clique has been disabled in this experiment.

Clicking on a listed clique displays its membership details. The membership of a sample clique is illustrated in Figure A.4(b). Clicking on individual members of the clique provides further information. All members of the clique are considered to be trusted contacts within the CTN.

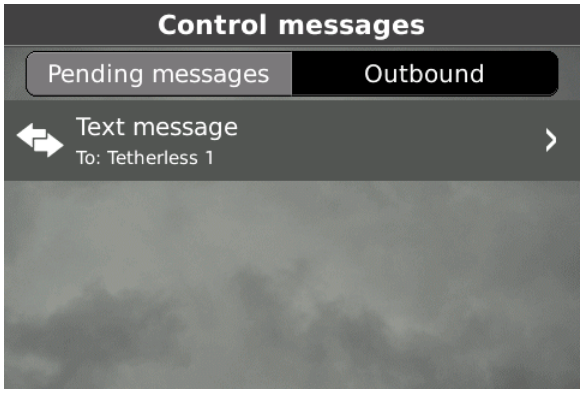


Figure A.5: MobiTether screen for displaying a list of pending control messages.

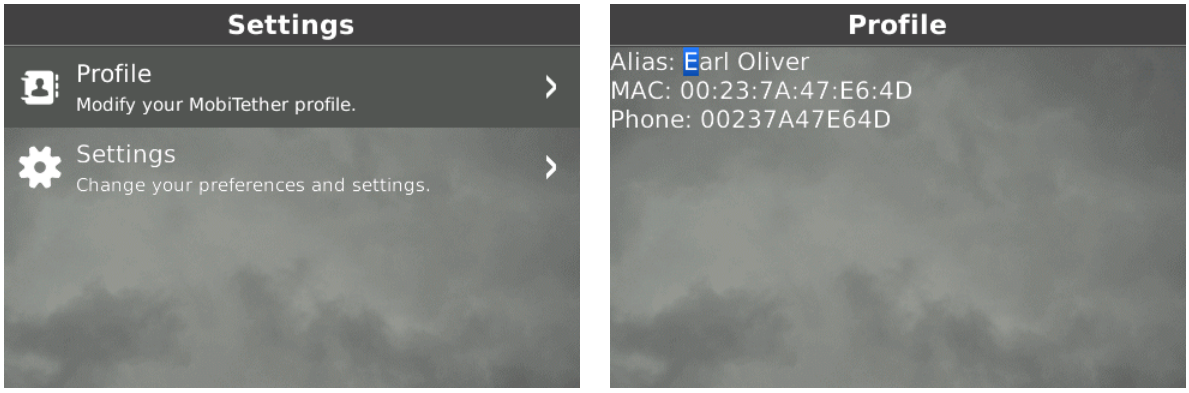
Selecting a contact reveals an option in the menu to `Send text message`. Text messages exchanged between users are control messages. In practice, these messages are intended to allow direct communication and possibly reduce the delay in retrieving desired content. Figures A.4(c) and A.4(d) illustrate this.

A.1.4 Control Messages

This system uses an SMS-based control channel to reduce the time required to propagate clique membership state to member contacts. Messages exchanged over the control channel are temporarily visible in the `Outbox`, illustrated in Figure A.5. Messages received from other devices are listed in `Pending messages`. In the current implementation, there are only two forms of actionable messages: text messages received from other users, and clique membership requests.

A.1.5 Settings

The Settings screen allows users to modify both their profile in the system and operating settings. The settings process is illustrated in Figure A.6(a) and Figure A.6(b).



(a) Alter application settings.

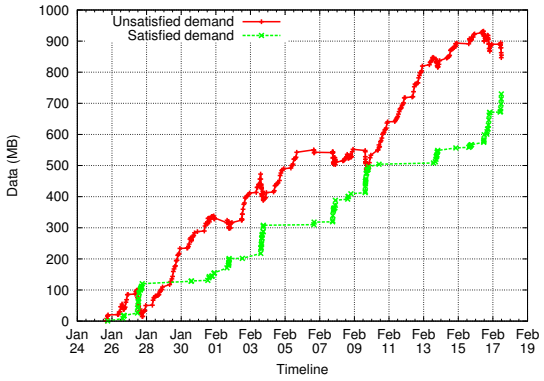
(b) Alter the alias displayed to other clique members.

Figure A.6: MobiTether screens for changing settings and profile information.

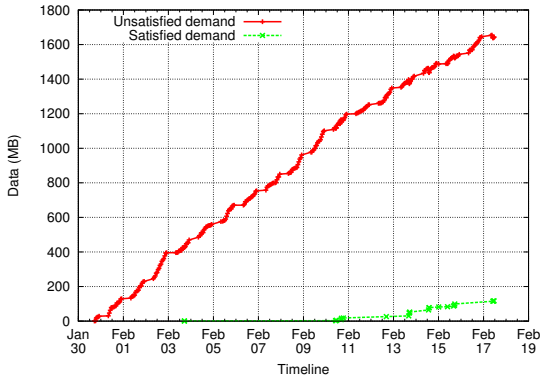
Appendix B

Outstanding and Satisfied Demand Traces for Field Trial Nodes

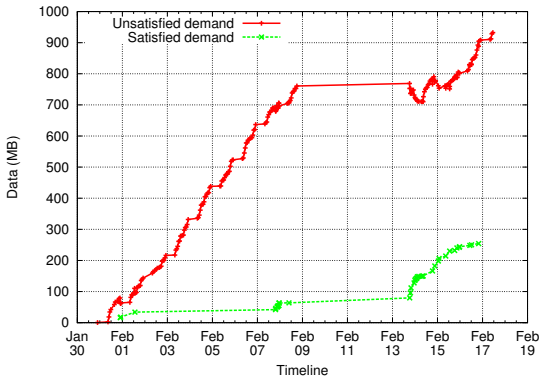
This appendix contains the traces of satisfied and unsatisfied demand for each node in the field trial described in Chapter 5. Many nodes in the field trial are highly connected and their demand for content is steadily satisfied throughout the field trial. Many other nodes are poorly connected and retrieve very little demanded content. The purpose of this appendix is to illustrate this variation.



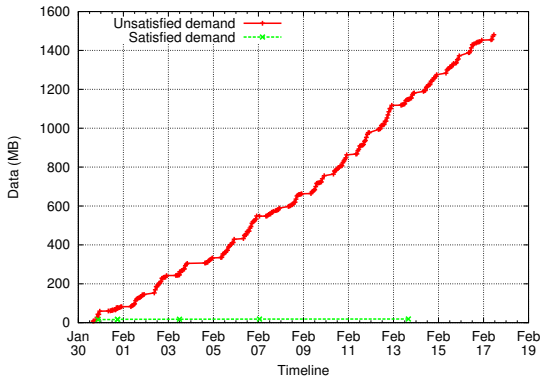
(a) Node 1.



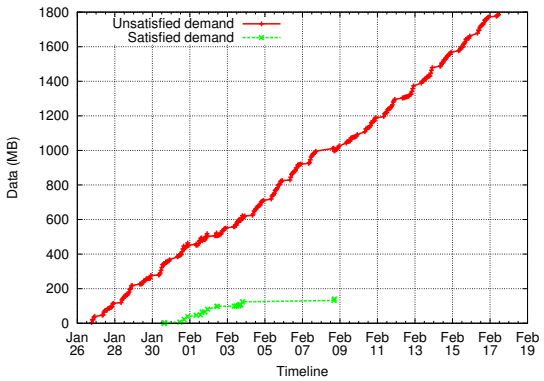
(b) Node 2.



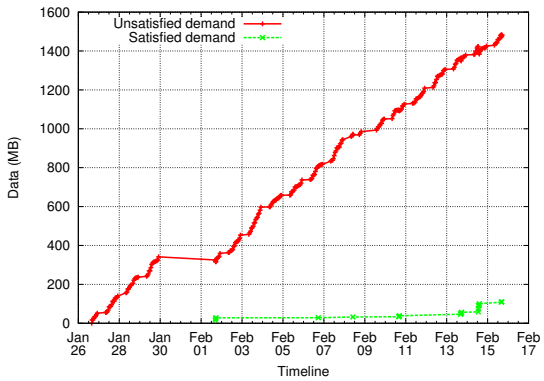
(c) Node 3.



(d) Node 4.

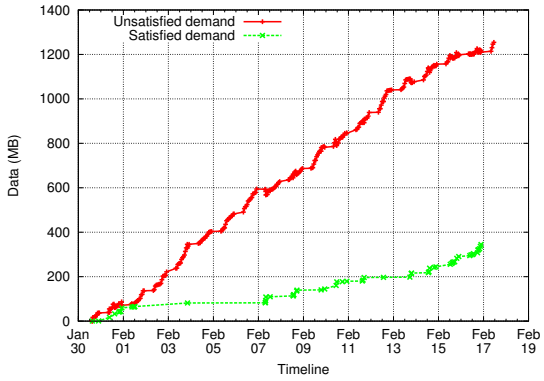


(e) Node 5.

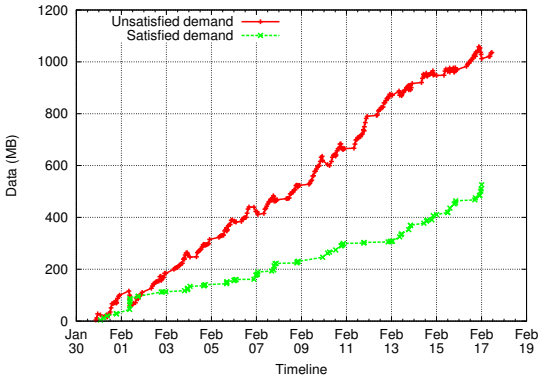


(f) Node 6.

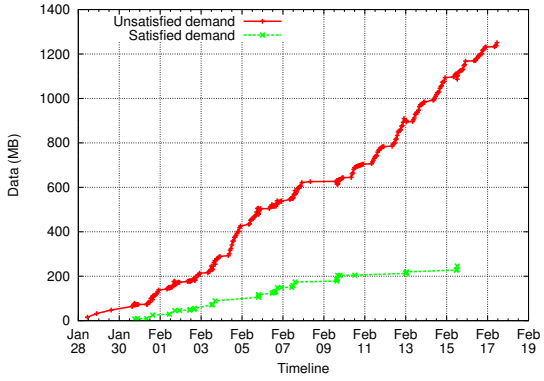
Figure B.1: Satisfied and unsatisfied demand for nodes 1 - 6.



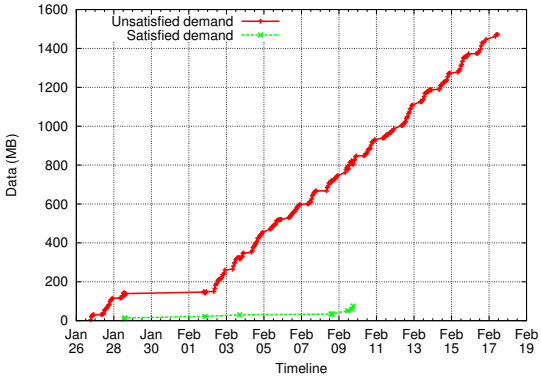
(a) Node 7.



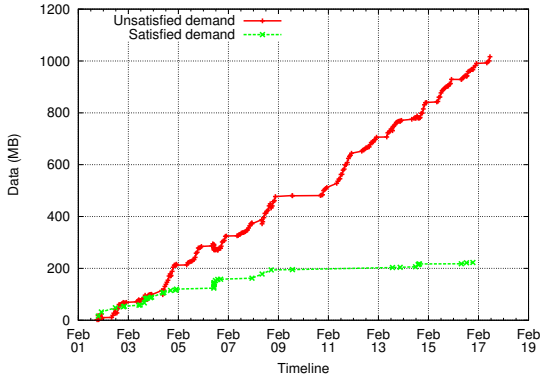
(b) Node 8.



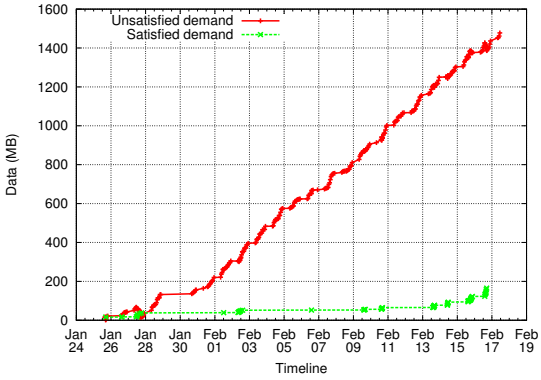
(c) Node 9.



(d) Node 10.

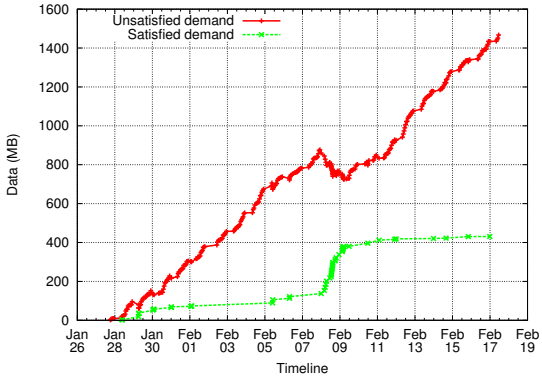


(e) Node 11.

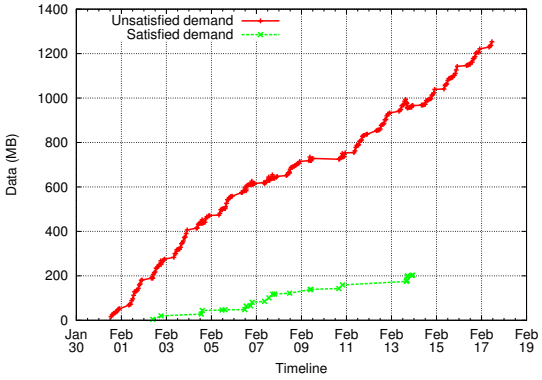


(f) Node 12.

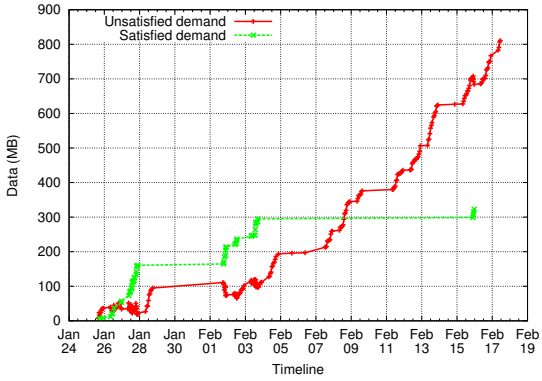
Figure B.2: Satisfied and unsatisfied demand for nodes 7 - 12.



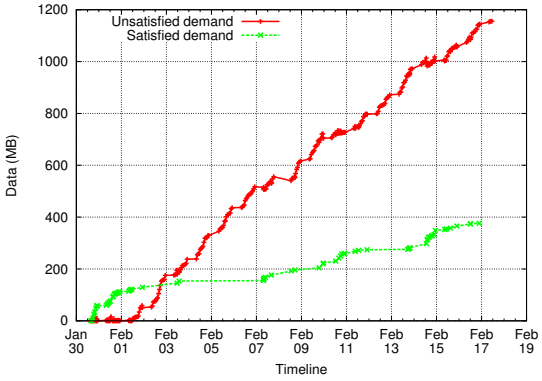
(a) Node 13.



(b) Node 14.

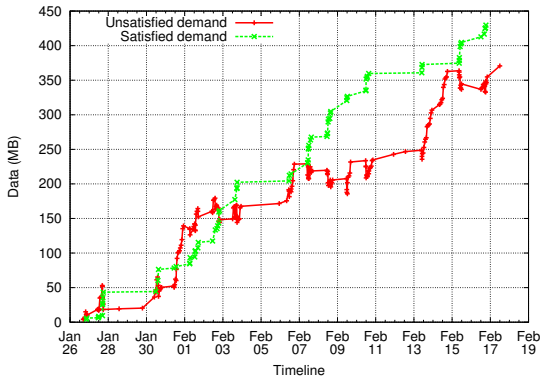


(c) Node 15: No content demand was satisfied between February 4 and February 16.

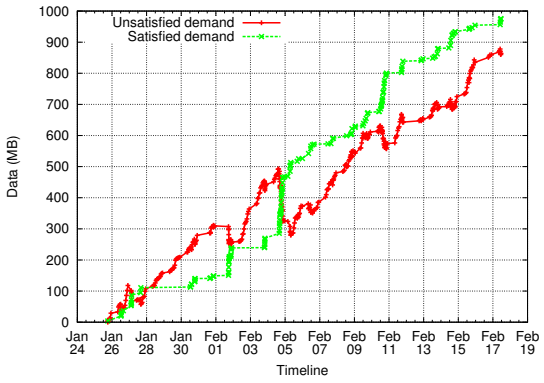


(d) Node 16.

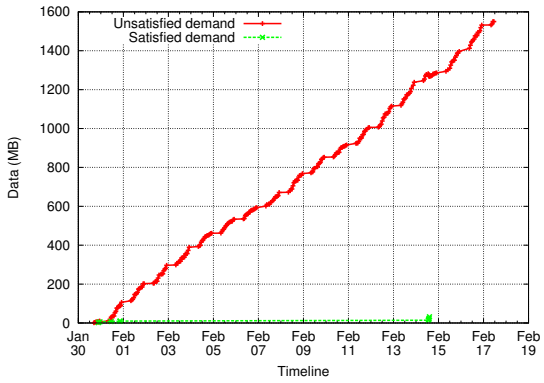
Figure B.3: Satisfied and unsatisfied demand for nodes 13 - 16.



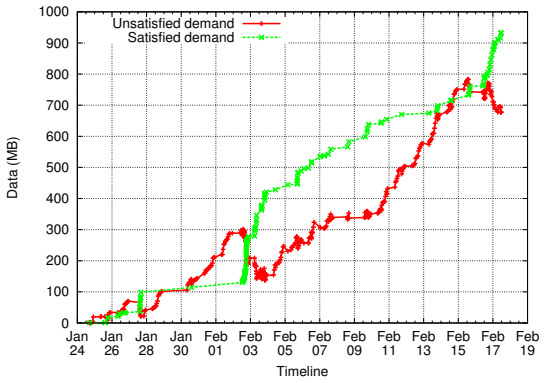
(a) Node 17.



(b) Node 18: Approximately 200 MB of demand was satisfied on the evening of February 4.

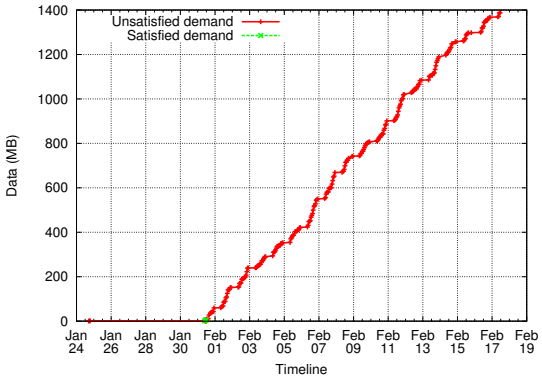


(c) Node 19: No demand was satisfied between February 1 and February 14.



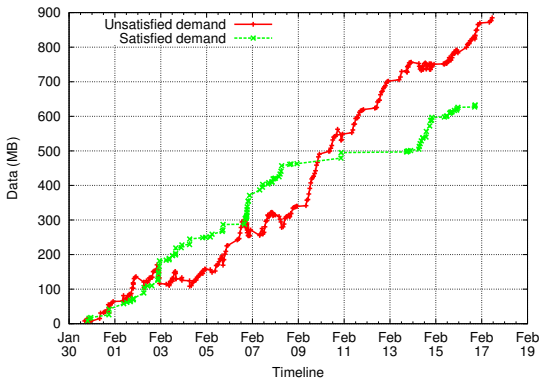
(d) Node 20: Node had poor connectivity before February 2. Approximately 300 MB of demand was subsequently satisfied causing the total demand satisfied to exceed the total unsatisfied demand.

Figure B.4: Satisfied and unsatisfied demand for nodes 17 - 20.

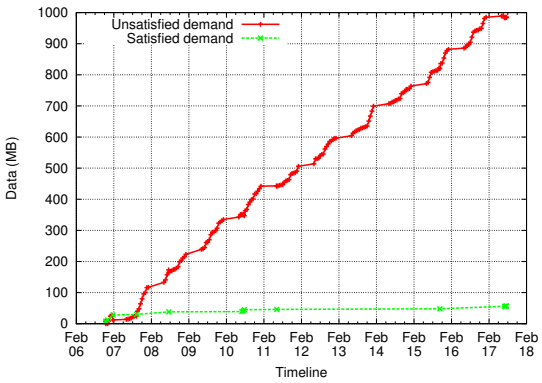


(a) Node 21 (no demand expressed).

(b) Node 22.

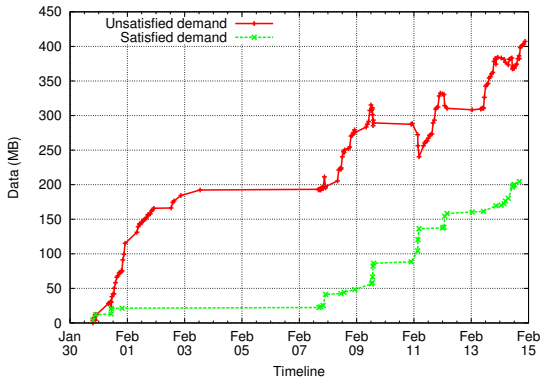


(c) Node 23. Node had poor connectivity between February 9 and February 14, which caused the total unsatisfied demand to exceed total satisfied demand.



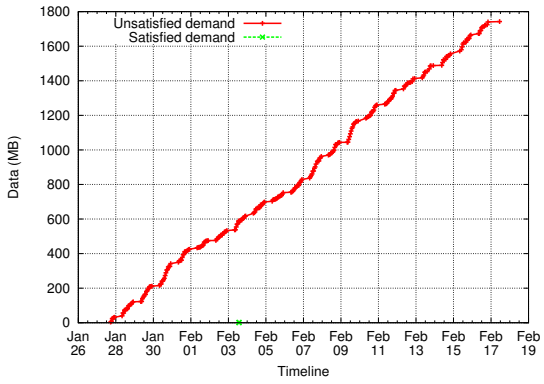
(d) Node 24.

Figure B.5: Satisfied and unsatisfied demand for nodes 21 - 24.

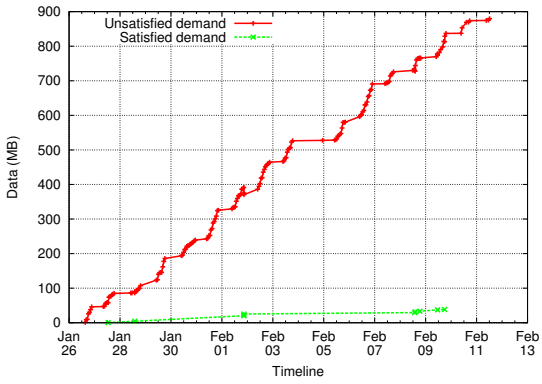


(a) Node 25. No demand was satisfied between February 2 and February 7.

(b) Node 26 (no demand expressed).

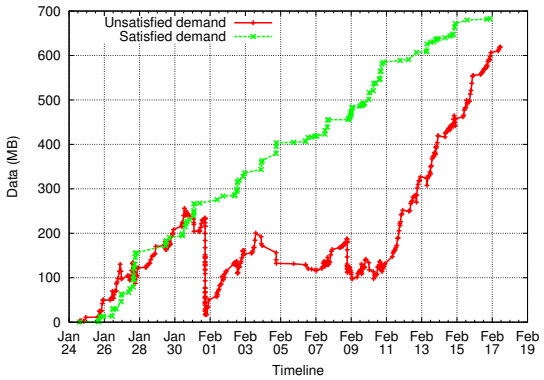


(c) Node 27. Demand was only satisfied on February 3.

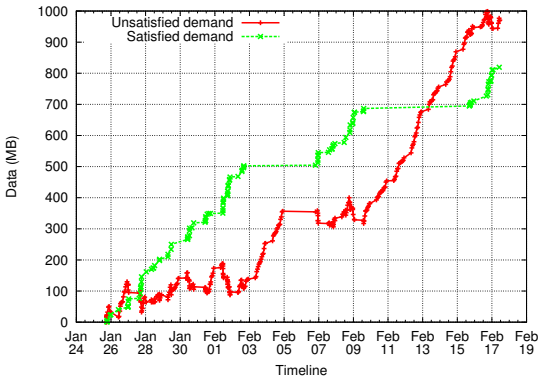


(d) Node 28.

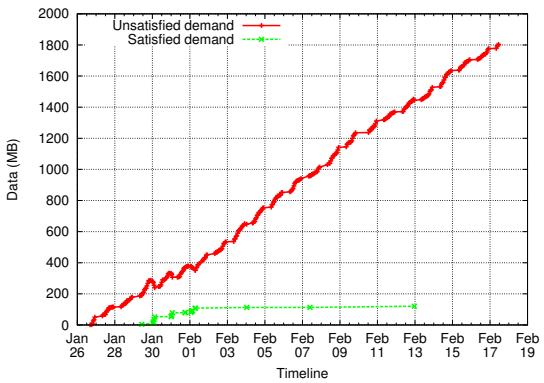
Figure B.6: Satisfied and unsatisfied demand for nodes 25 - 28.



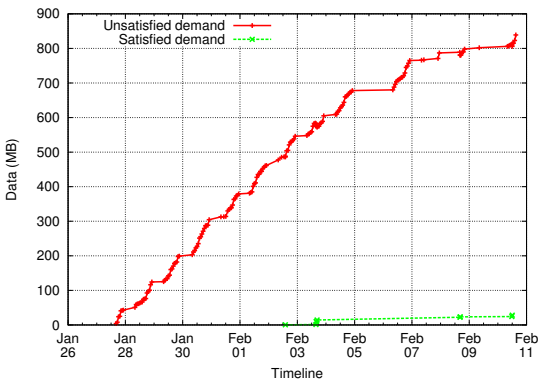
(a) Node 29. User manually withdrew demand for approximately 200 MB of content on January 31.



(b) Node 30.

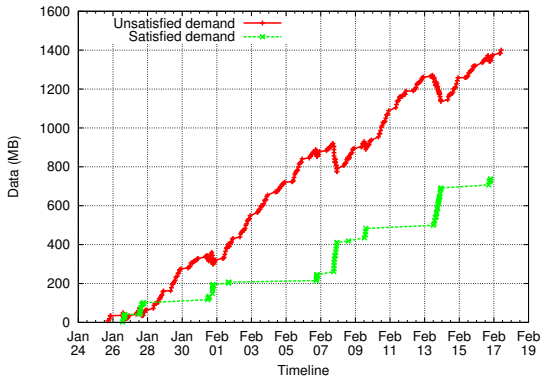


(c) Node 31.

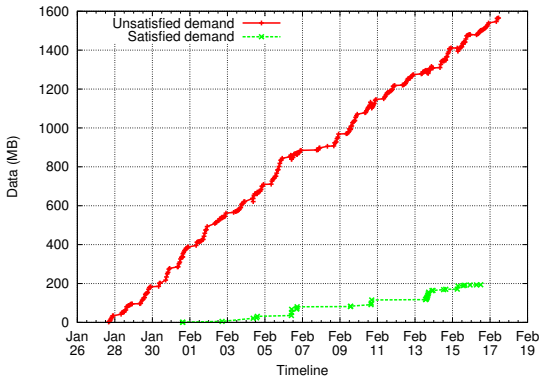


(d) Node 32: No demand was satisfied before February 2.

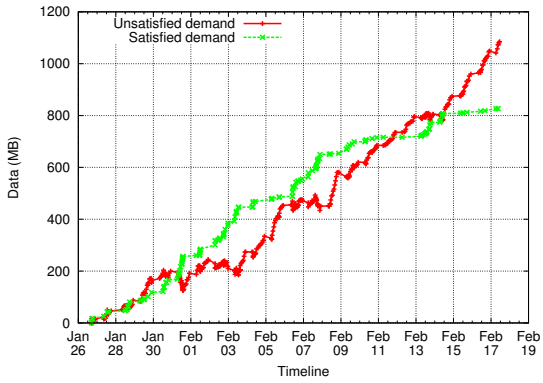
Figure B.7: Satisfied and unsatisfied demand for nodes 29 - 32.



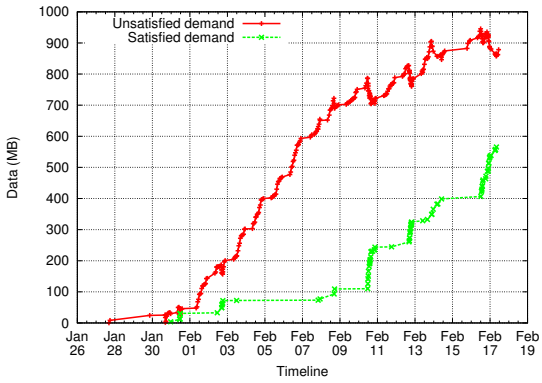
(a) Node 33.



(b) Node 34.



(c) Node 35.



(d) Node 36.

Figure B.8: Satisfied and unsatisfied demand for nodes 33 - 36.

References

[1] 3GPP Specification. http://www.3gpp.org/ftp/Specs/html-info/23040.htm, Last visited: February 12, 2012.

[2] Android security overview. http://source.android.com/tech/security/, Last visited: February 12, 2012.

[3] Apple. http://www.apple.com/, Last visited: February 12, 2012.

[4] Gammu. http://gammu.org, Last visited: February 12, 2012.

[5] Microsoft Corporation. http://www.microsoft.com, Last visited: February 12, 2012.

[6] Research in Motion Ltd. http://www.rim.com, Last visited: February 12, 2012.

[7] Rogers Communications Inc. http://www.rogers.com, Last visited: February 12, 2012.

[8] The Great Firewall: China's Web Users Battle Censorship, April 2010. http://www.time.com/time/world/article/0,8599,1981566,00.html, Last visited: April 15, 2012.

[9] Acision. Huge global growth in SMS continues over the new year period, January 2008. http://www.acision.com/news-and-events/press-releases/all-destinations/2008/huge-global-growth-in-sms-continues-over-the-new-year-period.aspx, Last visited: February 12, 2012.

[10] Adisasta Software. WinMobile Torrent. `http://www.wmtorrent.com/`, Last visited: March 20, 2012.

[11] Y. Agarwal, R. Gupta, and C. Schurgers. Dynamic power management using on demand paging for networked embedded systems. In *Design Automation Conference, 2005. Proceedings of the ASP-DAC 2005. Asia and South Pacific*, volume 2, 2005.

[12] Yuvraj Agarwal, Ranveer Chandra, Alec Wolman, Paramvir Bahl, Kevin Chin, and Rajesh Gupta. Wireless wakeups revisited: energy management for voip over wi-fi smartphones. In *MobiSys '07*, pages 179–191, New York, NY, USA, 2007. ACM.

[13] Ganesh Ananthanarayanan, Venkata N. Padmanabhan, Lenin Ravindranath, and Chandramohan A. Thekkath. Combine: leveraging the power of wireless peers through collaborative downloading. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 286–298, New York, NY, USA, 2007. ACM.

[14] Ross J. Anderson and Fabien A.P. Petitcolas. On the limits of steganography. *Selected Areas in Communications, IEEE Journal on*, 16(4):474 –481, may 1998.

[15] C. Andren, T. Bozych, B. Rood, and D. Schultz. Prism power management modes. *Intersil Americas Inc. Application Note*, February 1997.

[16] Adrian Andronache, Matthias R. Brust, and Steffen Rothkugel. Hycast- podcast discovery in mobile networks. In *Proceedings of the 3rd ACM workshop on Wireless multimedia networking and performance modeling*, WMuNeP '07, pages 27–34, New York, NY, USA, 2007. ACM.

[17] AsiaNews. Internet censorship tightening in Vietnam, June 2010. `http://www.asianews.it/news-en/Internet-censorship-tightening-in-Vietnam-18746.html`, Last visited: April 15, 2012.

[18] AsiaNews.it. Beijing to monitor public Internet users, July 2011. `http://www.asianews.it/news-en/Beijing-to-monitor-public-Internet-users-22225.html`, Last visited: April 19, 2012.

[19] N. Banerjee, A. Rahmati, M.D. Corner, S. Rollins, and L. Zhong. Users and Batteries: Interactions and Adaptive Energy Management in Mobile Systems. *Lecture Notes in Computer Science*, 4717:217, 2007.

[20] Paul Bao and Xiaohu Ma. Mp3-resistant music steganography based on dynamic range transform. In *Intelligent Signal Processing and Communication Systems, 2004. ISPACS 2004. Proceedings of 2004 International Symposium on*, pages 266 – 271, nov. 2004.

[21] Frank Bellosa. The benefits of event: driven energy accounting in power-sensitive systems. In *Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*, EW 9, pages 37–42, New York, NY, USA, 2000. ACM.

[22] Ian Black. Saudia Arabia leads Arab regimes in Internet censorship, June 2009. `http://www.guardian.co.uk/world/2009/jun/30/internet-censorship-arab-regimes`, Last visited: April 1, 2012.

[23] D. Bohman, M. Frank, P. Martini, and C. Scholz. Performance of symmetric neighbor discovery in bluetooth ad hoc networks. In *German Workshop on Mobile Ad-hoc Networking (WMAN04)*, 2004.

[24] Vincent Borrel, Mostafa H. Ammar, and Ellen W. Zegura. Understanding the wireless and mobile network space: a routing-centered classification. In *Proceedings of the second ACM workshop on Challenged networks*, CHANTS '07, pages 11–18, New York, NY, USA, 2007. ACM.

[25] Eric Brewer, Michael Demmer, Bowei Du, Melissa Ho, Matthew Kam, Sergiu Nedevschi, Joyojeet Pal, Rabin Patra, Sonesh Surana, and Kevin Fall. The case for technology in developing regions. *Computer*, 38(6):25–38, June 2005.

[26] B. Burns, O. Brock, and B.N. Levine. Mv routing and capacity building in disruption tolerant networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 1, pages 398 – 408 vol. 1, march 2005.

[27] Cameron Camp. Iranian TOR "arms race" a shadow of things to come?, March 2012. http://blog.eset.com/2012/02/16/iranian-tor-arms-race-a-shadow-of-things-to-come/, Last visited: April 1, 2012.

[28] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. Splitstream: high-bandwidth multicast in cooperative environments. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 298–313, New York, NY, USA, 2003. ACM.

[29] Augustin Chaintreau, Pan Hui, Jon Crowcroft, Christophe Diot, Richard Gass, and James Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6(6):606–620, 2007.

[30] Bin Bin Chen and Mun Choon Chan. Mobicent: a credit-based incentive system for disruption tolerant network. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 –9, march 2010.

[31] 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.

[32] KIM Cheolgi, MA Joongsoo, and LEE Joonwon. A random inquiry procedure using Bluetooth. *IEICE Transactions on Communications*, 86(9):2672–2683, 2003.

[33] Yang-hua Chu and Hui Zhang. Considering altruism in peer-to-peer internet streaming broadcast. In *Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video*, NOSSDAV '04, pages 10–15, New York, NY, USA, 2004. ACM.

[34] Todd L. Cignetti, Kirill Komarov, and Carla Schlatter Ellis. Energy estimation tools for the palm. In *MSWIM '00: Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 96–103, New York, NY, USA, 2000. ACM.

[35] T.C. Clancy and B.D. Walker. Meshtest: Laboratory-based wireless testbed for large topologies. In *TridentCom 2007.*, pages 1–6. IEEE, 2008.

[36] D. D. Clark, M. L. Lambert, and L. Zhang. NETBLT: a high throughput transport protocol. *SIGCOMM Computer Communication Review*, 17(5):353–359, 1987.

[37] D.D. Clark, M.L. Lambert, and L. Zhang. NETBLT: a bulk data transfer protocol. RFC 998, March 1987.

[38] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore Hong. Freenet: A distributed anonymous information storage and retrieval system. In Hannes Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *Lecture Notes in Computer Science*, pages 46–66. Springer Berlin / Heidelberg, 2001. 10.1007/3-540-44702-4.

[39] B. Cohen. Incentives Build Robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems*, volume 6. Berkeley, CA, USA, 2003.

[40] Margarita Valdés Cortés. Internet censorship around the world. `http://www.isoc.org/inet2000/cdproceedings/8k/8k_4.htm`, Last visited: April 1, 2012.

[41] P. Costa, C. Mascolo, M. Musolesi, and G.P. Picco. Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks. *Selected Areas in Communications, IEEE Journal on*, 26(5):748–760, 2008.

[42] Ruben Cuevas, Michal Kryczka, Angel Cuevas, Sebastian Kaune, Carmen Guerrero, and Reza Rejaie. Is content publishing in bittorrent altruistic or profit-driven? In *Proceedings of the 6th International Conference*, Co-NEXT '10, pages 11:1–11:12, New York, NY, USA, 2010. ACM.

[43] T. Dierks and E. Rescorla. RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2. Technical report, August 2008.

[44] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: the second-generation onion router. In *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[45] A. Doria, M. Uden, and D. Pandey. Providing connectivity to the saami nomadic community. volume 1, page 3. Citeseer, 2002.

[46] C. Drula. Fast and Energy Efficient Neighbour Discovery for Opportunistic Networking with Bluetooth. Master's thesis, University of Toronto, 2005.

[47] C. Drula, C. Amza, F. Rousseau, and A. Duda. Adaptive Energy Conserving Algorithms for Neighbor Discovery in Opportunistic Bluetooth Networks. *IEEE Journal on Selected Areas in Communications*, 25(1):96, 2007.

[48] William Enck, Patrick Traynor, Patrick McDaniel, and Thomas La Porta. Exploiting open functionality in SMS-capable cellular networks. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, pages 393–404, New York, NY, USA, 2005. ACM.

[49] ETSI. Technical realization of the short message service. TS 100 901, GSM 03.40 version 6.1.0, July 1998. `http://www.3gpp.org/ftp/Specs/html-info/23040.htm`, Last visited: February 12, 2012.

[50] EzTexting.com. 50 fantastic uses of SMS text messaging. `http://www.eztexting.com/uses-of-text-messaging.html`, Last visited: April 1, 2012.

[51] Hossein Falaki, Ratul Mahajan, and Deborah Estrin. Systemsens: a tool for monitoring usage in smartphone research deployments. In *Proceedings of the sixth international workshop on MobiArch*, MobiArch '11, pages 25–30, New York, NY, USA, 2011. ACM.

[52] Hossein Falaki, Ratul Mahajan, Srikanth Kandula, Dimitrios Lymberopoulos, Ramesh Govindan, and Deborah Estrin. Diversity in smartphone usage. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 179–194, New York, NY, USA, 2010. ACM.

[53] M.H. Falaki. WLAN Interface Management on Mobile Devices. Master's thesis, University of Waterloo, Canada, Ontario, August 2008.

[54] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, archi-*

tectures, and protocols for computer communications, pages 27–34, New York, NY, USA, 2003. ACM.

[55] Amos Fiat and Jared Saia. Censorship resistant peer-to-peer content addressable networks. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '02, pages 94–103, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

[56] L. Galluccio, G. Morabito, and S. Palazzo. Analytical evaluation of a tradeoff between energy efficiency and responsiveness of neighbor discovery in self-organizing ad hoc networks. *Selected Areas in Communications, IEEE Journal on*, 22(7):1167–1182, 2004.

[57] 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.

[58] Eric Gilbert and Karrie Karahalios. Predicting tie strength with social media. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 211–220, New York, NY, USA, 2009. ACM.

[59] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. YouTube traffic characterization: a view from the edge. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 15–28, New York, NY, USA, 2007. ACM.

[60] Philippe Golle, Kevin Leyton-Brown, Ilya Mironov, and Mark Lillibridge. Incentives for sharing in peer-to-peer networks. In Ludger Fiege, Gero Mhl, and Uwe Wilhelm, editors, *Electronic Commerce*, volume 2232 of *Lecture Notes in Computer Science*, pages 75–87. Springer Berlin / Heidelberg, 2001.

[61] M.C. González, C.A. Hidalgo, and A.L. Barabási. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.

[62] Matthias Grossglauser and David N. C. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Trans. Netw.*, 10(4):477–486, 2002.

[63] GSM World. GSM Coverage Maps. `http://www.gsmworld.com/roaming/gsminfo/index.shtml`, Last visited: February 12, 2012.

[64] Saikat Guha and Paul Francis. Simple traversal of udp through nats and tcp too (stunt). `http://nutss.gforge.cis.cornell.edu/stunt.php`, Last visited: February 12, 2012.

[65] Saikat Guha, Yutaka Takeda, and Paul Francis. Nutss: a sip-based approach to udp and tcp network connectivity. In *Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, FDNA '04, pages 43–48, New York, NY, USA, 2004. ACM.

[66] S. Guo, M. Derakhshani, M.H. Falaki, U. Ismail, R. Luk, E.A. Oliver, S. Ur Rahman, A. Seth, M.A. Zaharia, and S. Keshav. Design and implementation of the KioskNet system. *Computer Networks*, 55(1):264 – 281, 2011.

[67] Shimin Guo, Mohammad Hossein Falaki, Earl A. Oliver, Sumair Ur Rahman, Aaditeshwar Seth, Matei A. Zaharia, Usman Ismail, and Srinivasan Keshav. Design and implementation of the kiosknet system. In *Proceedings of the International Conference on Information and Communication Technologies and Development (ICTD 2007)*, pages 300–309, 2007.

[68] Shimin Guo and Srinivasan Keshav. Fair and efficient scheduling in data ferrying networks. In *CoNEXT '07: Proceedings of the 2007 ACM CoNEXT conference*, pages 1–12, New York, NY, USA, 2007. ACM.

[69] J. Haartsen. Bluetooth-The universal radio interface for ad hoc, wireless connectivity. *Ericsson Review*, 3(1):110–117, 1998.

[70] David Hadaller, Srinivasan Keshav, Tim Brecht, and Shubham Agarwal. Vehicular opportunistic communication under the microscope. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 206–219, New York, NY, USA, 2007. ACM.

[71] Yieh-Ran Haung and Jan-Ming Ho. Overload control for short message transfer in GPRS/UMTS networks. *Information Sciences*, 170(2-4):235–249, 2005.

[72] 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.

[73] Ólafur R. Helgason, Emre A. Yavuz, Sylvia T. Kouyoumdjieva, Ljubica Pajevic, and Gunnar Karlsson. A mobile peer-to-peer system for opportunistic content-centric networking. In *Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds*, MobiHeld '10, pages 21–26, New York, NY, USA, 2010. ACM.

[74] Amer Hildebrandt. An unintentional martyr: Neda becomes 'symbol of goodness', June 2009. `http://www.cbc.ca/news/world/story/2009/06/22/f-neda-iran.html`, Last visited: April 1, 2012.

[75] P. Hui, A. Chaintreau, R. Gass, J. Scott, J. Crowcroft, and C. Diot. Pocket Switched Networking: Challenges, Feasibility and Implementation Issues. *Lecture Notes in Computer Science*, 3854:1, 2006.

[76] Pan Hui, Augustin Chaintreau, James Scott, Richard Gass, Jon Crowcroft, and Christophe Diot. Pocket switched networks and human mobility in conference environments. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 244–251, New York, NY, USA, 2005. ACM.

[77] Pan Hui, Jon Crowcroft, and Eiko Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 241–250, New York, NY, USA, 2008. ACM.

[78] Informa Telecoms & Media. Global SMS Traffic to Reach 8.7 Trillion in 2015, January 2011. `http://www.informatm.com/itmgcontent/icoms/whats-new/20017843617.html`, Last visited: March 20, 2012.

[79] International Freedom of Expression Exchange. Journalists and activists agitating for rights targeted, August 2011. `http://www.ifex.org/cambodia/2011/08/10/artivists_targeted/`, Last visited: April 15, 2012.

[80] Sushant Jain, Kevin Fall, and Rabin Patra. Routing in a delay tolerant network. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '04, pages 145–158, New York, NY, USA, 2004. ACM.

[81] Java Community Process. JSR 120, Wireless Messaging API. `http://jcp.org/en/jsr/detail?id=120`, Last visited: February 12, 2012.

[82] Neil. Johnson and Sushil Jajodia. Exploring steganography: Seeing the unseen. *Computer*, 31(2):26 –34, feb. 1998.

[83] Evan Jones and Paul Ward. Routing strategies for delay-tolerant networks. 2006.

[84] Evan P. C. Jones, Lily Li, and Paul A. S. Ward. Practical routing in delay-tolerant networks. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 237–243, New York, NY, USA, 2005. ACM.

[85] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. In *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, ASPLOS-X, pages 96–107, New York, NY, USA, 2002. ACM.

[86] Sewook Jung, Uichin Lee, Alexander Chang, Dae-Ki Cho, and Mario Gerla. Bluetorrent: Cooperative content sharing for bluetooth users. *Pervasive Mob. Comput.*, 3(6):609–634, 2007.

[87] Matjaz B. Juric, Ivan Rozman, Bostjan Brumen, Matjaz Colnaric, and Marjan Hericko. Comparison of performance of web services, ws-security, rmi, and rmissl. *Journal of Systems and Software*, 79(5):689 – 700, 2006. ¡ce:title¿Quality Software¡/ce:title¿.

[88] Michael Kan. China Demands Real Names From Mobile Phone Users, September 2010. `http://www.pcworld.com/businesscenter/article/204616/china_demands_real_names_from_mobile_phone_users.html`, Last visited: April 1, 2012.

[89] Marko Kananen. Shadow Internet Freedom or Control?, June 2011. http://www.thebeginner.eu/technology/all-in-innovation/538-shadow-internet--freedom-or-control, Last visited: April 1, 2012.

[90] T. Karagiannis, J.Y. Le Boudec, and M. Vojnovic. Power law and exponential decay of intercontact times between mobile devices. *Mobile Computing, IEEE Transactions on*, 9(10):1377–1390, 2010.

[91] T. Kathiravelu and A. Pears. What & When?: Distributing Content in Opportunistic Networks. In *Proceedings of the International Conference on Wireless and Mobile Computing (ICWMC 2006)*.

[92] S. Keshav. Why cell phones will dominate the future internet. *SIGCOMM Computing Communications Review*, 35(2):83–86, 2005.

[93] 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.

[94] Mugo Kibati and Donyaprueth Krairit. The wireless local loop in developing regions. *Commun. ACM*, 42(6):60–66, 1999.

[95] Dongkyun Kim, J. J. Garcia-Luna-Aceves, Katia Obraczka, Juan-Carlos Cano, and Pietro Manzoni. Routing mechanisms for mobile ad hoc networks based on the energy drain rate. *IEEE Transactions on Mobile Computing*, 2(2):161–173, 2003.

[96] Kanishka Lahiri, Sujit Dey, Debashis Panigrahi, and Anand Raghunathan. Battery-driven system design: A new frontier in low power design. In *ASP-DAC '02*, page 261, Washington, DC, USA, 2002. IEEE Computer Society.

[97] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, et al. Place Lab: Device Positioning Using Radio Beacons in the Wild. In *Proceedings of Pervasive*, volume 3468, pages 116–133. Springer, 2005.

[98] R. Lambiotte and M. Ausloos. Uncovering collective listening habits and music genres in bipartite networks. *Phys. Rev. E*, 72(6):066107, December 2005.

[99] G. Le Bodic. *Mobile messaging technologies and services: SMS, EMS and MMS*. John Wiley & Sons Inc, 2005.

[100] Jason LeBrun and Chen-Nee Chuah. Bluetooth content distribution stations on public transit. In *MobiShare '06: Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*, pages 63–65, New York, NY, USA, 2006. ACM.

[101] U. Lee, S. Jung, A. Chang, D.K. Cho, and M. Gerla. Bluetooth-based P2P Content Distribution to Mobile Users. Technical report, University of California, Los Angeles, 2007.

[102] Uichin Lee, Joon-Sang Park, Joseph Yeh, Giovanni Pau, and Mario Gerla. Code torrent: content distribution using network coding in vanet. In *MobiShare '06: Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*, pages 1–5, New York, NY, USA, 2006. ACM.

[103] Jérémie Leguay, Timur Friedman, and Vania Conan. Dtn routing in a mobility pattern space. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, WDTN '05, pages 276–283, New York, NY, USA, 2005. ACM.

[104] Jérémie Leguay, Anders Lindgren, James Scott, Timur Friedman, and Jon Crowcroft. Opportunistic content distribution in an urban setting. In *CHANTS '06: Proceedings of the 2006 SIGCOMM workshop on Challenged networks*, pages 205–212, New York, NY, USA, 2006. ACM.

[105] Vincent Lenders, Martin May, Gunnar Karlsson, and Clemens Wacha. Wireless ad hoc podcasting. *SIGMOBILE Mobile Computing Communication Review*, 12(1):65–67, January 2008.

[106] Q. Li, S. Zhu, and G. Cao. Routing in socially selfish delay tolerant networks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[107] Marc Liberatore, Brian Neil Levine, and Chadi Barakat. Maximizing transfer opportunities in bluetooth dtns. In *CoNEXT '06: Proceedings of the 2006 ACM CoNEXT conference*, pages 1–11, New York, NY, USA, 2006. ACM.

[108] Anders Lindgren, Avri Doria, and Olov Scheln. Probabilistic routing in intermittently connected networks. In Petre Dini, Pascal Lorenz, and Jos de Souza, editors, *Service Assurance with Partial and Intermittent Resources*, volume 3126 of *Lecture Notes in Computer Science*, pages 239–254. Springer Berlin / Heidelberg, 2004.

[109] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer. Free riding in bittorrent is cheap. *Connections*, 300:400.

[110] Austin Mackell. Alber Saber Arrested for Atheism, September 2012. `https://austingmackell.wordpress.com/2012/09/16/alber-samer-arrested-for-atheism/`, Last visited: September 20, 2012.

[111] Natalia Marmasse and Chris Schmandt. A user-centered location model. *Personal Ubiquitous Computing*, 6(5-6):318–321, 2002.

[112] Martin May, Vincent Lenders, Gunnar Karlsson, and Clemens Wacha. Wireless opportunistic podcasting: implementation and design tradeoffs. In *Proceedings of the second ACM workshop on Challenged networks*, CHANTS '07, pages 75–82, New York, NY, USA, 2007. ACM.

[113] Liam McNamara, Cecilia Mascolo, and Licia Capra. Media sharing based on colocation prediction in urban transport. In *MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 58–69, New York, NY, USA, 2008. ACM.

[114] 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.

[115] X. Meng, P. Zerfos, V. Samanta, SHY Wong, and S. Lu. Analysis of the Reliability of a Nationwide Short Message Service. *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1811–1819, 2007.

[116] A.G. Miklas, K.K. Gollu, K.K.W. Chan, S. Saroiu, K.P. Gummadi, and E. de Lara. Exploiting Social Interactions in Mobile Systems. *Lecture Notes in Computer Science*, 4717:409, 2007.

[117] Stanley Milgram. *The Individual in a Social World: Essays and Experiments*. Addison-Wesley Publishing Company, Reading, MA, 1977.

[118] Abderrahmen Mtibaa, Augustin Chaintreau, Jason LeBrun, Earl Oliver, Anna-Kaisa Pietiläinen, and Christophe Diot. Are you moved by your social network application? In *WOSN '08: Proceedings of the first workshop on online social networks*, pages 67–72, New York, NY, USA, 2008. ACM.

[119] P. Murphy, E. Welsh, and J.P. Frantz. Using bluetooth for short-term ad hoc connections between moving vehicles: A feasibility study. In *IEEE Vehicular Technology Conference*, volume 1, pages 414–418, 2002.

[120] Z. Naor. An efficient short messages transmission in cellular networks. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 4 vol. (xxxv+2866), March 2004.

[121] Rolf Neugebauer and Derek McAuley. Energy is just another resource: Energy accounting and energy pricing in the nemesis os. In *HOTOS '01*, page 67, Washington, DC, USA, 2001. IEEE Computer Society.

[122] NBC News. Syria activists using us tech to beat curbs, June 2012. `http://www.msnbc.msn.com/id/47905226/ns/technology_and_science-security/t/syria-activists-using-us-tech-beat-curbs/#.UHX5pLQvEUU`, Last visited: June 21, 2012.

[123] Anthony J. Nicholson and Brian D. Noble. Breadcrumbs: forecasting mobile connectivity. In *MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 46–57, New York, NY, USA, 2008. ACM.

[124] Earl Oliver. SMS-TP Files. `http://blizzard.cs.uwaterloo.ca/eaoliver/sms.html`, Last visited: April 12, 2012.

[125] Earl Oliver. A survey of platforms for mobile networks research. *SIGMOBILE Moble Computer Communication Review*, 12(4):56–63, 2008.

[126] Earl Oliver. The challenges in large-scale smartphone user studies. In *HotPlanet '10: Proceedings of the 2nd ACM International Workshop on Hot Topics in Planet-scale Measurement*, pages 1–5, New York, NY, USA, 2010. ACM.

[127] Earl Oliver. Characterizing the transport behaviour of the short message service. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 223–238, New York, NY, USA, 2010. ACM.

[128] Earl Oliver and Hossein Falaki. Performance evaluation and analysis of delay tolerant networking. In *MobiEval '07: Proceedings of the 1st international workshop on System evaluation for mobile platforms*, pages 1–6, New York, NY, USA, 2007. ACM Press.

[129] Earl Oliver and Srinivasan Keshav. Design principles for opportunistic communication in constrained computing environments. In *WiNS-DR '08: Proceedings of the 2008 ACM workshop on Wireless networks and systems for developing regions*, pages 31–36, New York, NY, USA, 2008. ACM.

[130] Earl A. Oliver and Srinivasan Keshav. An empirical approach to smartphone energy level prediction. In *Proceedings of the 13th international conference on Ubiquitous computing*, UbiComp '11, pages 345–354, New York, NY, USA, 2011. ACM.

[131] Open Handset Alliance. Wi-Fi Direct — Android Developers, 2012. `http://developer.android.com/guide/topics/connectivity/wifip2p.html`, Last visited: April 1, 2012.

[132] J. Ott and D. Kutscher. Drive-thru Internet: IEEE 802.11 b for" automobile" users. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 1, 2004.

[133] V.N. Padmanabhan, H.J. Wang, and P.A. Chou. Resilient peer-to-peer streaming. In *Network Protocols, 2003. Proceedings. 11th IEEE International Conference on*, pages 16–27. IEEE, 2003.

[134] Tal Pavel. Blocking of Wikipedia reported in Iran, August 2010. `http://advocacy.globalvoicesonline.org/2010/08/04/blocking-of-wikipedia-reported-in-iran/`, Last visited: April 1, 2012.

[135] V. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California at Berkeley, Berkeley, CA, USA, 1998.

[136] 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.

[137] Alex (Sandy) Pentland, Richard Fletcher, and Amir Hasson. Daknet: Rethinking connectivity in developing nations. *Computer*, 37(1):78–83, 2004.

[138] Trevor Pering, Yuvraj Agarwal, Rajesh Gupta, and Roy Want. Coolspots: reducing the power consumption of wireless mobile devices with multiple radio interfaces. In *MobiSys '06: Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 220–232, New York, NY, USA, 2006. ACM.

[139] Trevor Pering, Vijay Raghunathan, and Roy Want. Exploiting radio hierarchies for power-efficient wireless device discovery and connection setup. In *VLSID '05: Proceedings of the 18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design*, pages 774–779, Washington, DC, USA, 2005. IEEE Computer Society.

[140] A. Pfitzmann and M. Hansen. Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management-a consolidated proposal for terminology. Technical report, Technische Universitat Dresden, February 2008.

[141] Michael Piatek, Tomas Isdal, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. Do incentives build robustness in bit torrent. In *Proceedings of the 4th USENIX conference on Networked systems design & implementation*, NSDI'07, pages 1–1, Berkeley, CA, USA, 2007. USENIX Association.

[142] A-K. Pietiläinen and C. Diot. Connectivity management for opportunistic communications in psn. Technical Report CR-PRL-2008-03-0001, Thomson, Paris, March 2008.

[143] Anna-Kaisa Pietiläinen, Earl Oliver, Jason LeBrun, George Varghese, Jon Crowcroft, and Christophe Diot. Experiments in mobile social networking. Technical Report CR-PRL-2008-02-0003, Thomson, February 2008.

[144] Anna-Kaisa Pietiläinen, Earl Oliver, Jason LeBrun, George Varghese, and Christophe Diot. Mobiclique: middleware for mobile social networking. In *Proceedings of the*

2nd ACM workshop on Online social networks, WOSN '09, pages 49–54, New York, NY, USA, 2009. ACM.

[145] Portio Research. Mobile Messaging Futures 2011 - 2015. `http://www.portioresearch.com/en/reports/current-portfolio/mobile-messaging-futures-2011-2015.aspx`, Last visited: April 5, 2012.

[146] Asfandyar Qureshi and John Guttag. Horde: separating network striping policy from mechanism. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 121–134, New York, NY, USA, 2005. ACM.

[147] Ahmad Rahmati, Angela Qian, and Lin Zhong. Understanding human-battery interaction on mobile phones. In *MobileHCI '07: Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, pages 265–272, New York, NY, USA, 2007. ACM.

[148] Ram Ramanathan, Richard Hansen, Prithwish Basu, Regina Rosales-Hain, and Rajesh Krishnan. Prioritized epidemic routing for opportunistic networks. In *MobiOpp '07: Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking*, pages 62–66, New York, NY, USA, 2007. ACM.

[149] N. Ravi, J. Scott, L. Han, and L. Iftode. Context-aware battery management for mobile phones. In *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, pages 224–233. IEEE, 2008.

[150] Reporters Without Borders. Internet enemies, March 2011. `http://12mars.rsf.org/2011/i/Internet_Enemies.pdf`, Last visited: April 15, 2012.

[151] Research in Motion Ltd. BlackBerry JDE 5.0.0 API reference, 2010. `http://www.blackberry.com/developers/docs/5.0.0api/index.html`, Last visited: April 1, 2012.

[152] Rezwan. Bangladesh: Court Orders Shutting Down of Facebook Pages for Blasphemous Contents, March 2012. `http://globalvoicesonline.org/2012/03/24/bangladesh-court-orders-shutting-down-of-facebook-pages-for-\blasphemous-contents/`, Last visited: April 15, 2012.

[153] Injong Rhee, Minsu Shin, Seongik Hong, Kyunghan Lee, and Song Chong. Human mobility patterns and their impact on routing in human-driven mobile networks. In *HotNets '07: Proceedings of the 2007 workshop on Hot Topics in Networks*, 2007.

[154] Christopher Rhoads and Geoffrey A. Fowler. Egypt Shuts Down Internet, Cellphone Services, January 2011. `http://online.wsj.com/article/SB10001424052748703956604576110453371369740.html`, Last visited: April 1, 2012.

[155] Michael Rogers. Censorship-Resistant Communication over Public Networks. 2006.

[156] M.A. Rónai and E. Kail. A simple neighbour discovery procedure for Bluetooth ad hoc networks. In *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE*, volume 2, 2003.

[157] A.N. Rosenberg and S. Kemp. *CDMA capacity and quality optimization*. McGraw-Hill Professional, 2003.

[158] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). RFC 3489, March 2003.

[159] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire. Distributed topology construction of Bluetooth wireless personal area networks. *Selected Areas in Communications, IEEE Journal on*, 23(3):633–643, 2005.

[160] Jochen H. Schiller. *Mobile communications*. Addison-Wesley, Reading, MA, USA, second edition, 2003.

[161] J. Scott, P. Hui, J. Crowcroft, and C. Diot. Haggle: A networking architecture designed around mobile users. *Proceedings of the Third Annual Conference on Wireless On demand Network Systems and Services (WONS 2006)*, 2006.

[162] I. Sedov, S. Preuss, C. Cap, M. Haase, and D. Timmermann. Time and energy efficient service discovery in Bluetooth. In *Vehicular Technology Conference, 2003. VTC 2003-Spring. The 57th IEEE Semiannual*, volume 1, 2003.

[163] A. Seth and S. Keshav. Practical security for disconnected nodes. In *Secure Network Protocols, 2005.(NPSec). 1st IEEE ICNP Workshop on*, pages 31–36. IEEE, 2005.

[164] A. Seth, M. Zaharia, S. Keshav, and S. Bhattacharyya. A policy-oriented architecture for opportunistic communication on multiple wireless networks. Technical report, University of Waterloo, 2006.

[165] Amol Sharma. Google, Facebook Fight Indian Censorship Demands, January 2012. `http://online.wsj.com/article/SB10001424052970204542404577158342623999990.html`, Last visited: April 15, 2012.

[166] Upendra Shevade, Han Hee Song, Lili Qiu, and Yin Zhang. Incentive-aware routing in DTNs. In *Network Protocols, 2008. ICNP 2008. IEEE International Conference on*, pages 238–247. IEEE, 2008.

[167] Eugene Shih, Paramvir Bahl, and Michael J. Sinclair. Wake on wireless: an event driven energy saving strategy for battery operated devices. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 160–171, New York, NY, USA, 2002. ACM.

[168] 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.

[169] Frank Siegemund and Michael Rohs. Rendezvous layer protocols for bluetooth-enabled smart devices. *Personal Ubiquitous Comput.*, 7(2):91–101, 2003.

[170] Inga Sikorskaya. Cyber-Censorship in Uzbekistan, March 20011. `http://iwpr.net/report-news/cyber-censorship-uzbekistan`, Last visited: April 15, 2012.

[171] Tara Small and Zygmunt J. Haas. Resource and performance tradeoffs in delay-tolerant wireless networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, WDTN '05, pages 260–267, New York, NY, USA, 2005. ACM.

[172] T. Spyropoulos, K. Psounis, and C.S. Raghavendra. Single-copy routing in intermittently connected mobile networks. In *Sensor and Ad Hoc Communications and*

Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on, pages 235–244. IEEE, 2005.

[173] Vikram Srinivasan, Mehul Motani, and Wei Tsang Ooi. Analysis and implications of student contact patterns derived from campus schedules. In *Proceedings of the 12th annual international conference on Mobile computing and networking*, MobiCom '06, pages 86–97, New York, NY, USA, 2006. ACM.

[174] J. Su, A. Chin, A. Popivanova, A. Goel, and E. de Lara. User mobility for opportunistic ad-hoc networking. In *Mobile Computing Systems and Applications, 2004. WMCSA 2004. Sixth IEEE Workshop on*, pages 41–50, 2004.

[175] J. Su, J. Scott, P. Hui, J. Crowcroft, E. de Lara, C. Diot, A. Goel, M.H. Lim, and E. Upton. Haggle: Seamless Networking for Mobile Applications.

[176] Jing Su, James Scott, Pan Hui, Jon Crowcroft, Eyal De Lara, Christophe Diot, Ashvin Goel, Meng How Lim, and Eben Upton. Haggle: seamless networking for mobile applications. In *Proceedings of the 9th international conference on Ubiquitous computing*, UbiComp '07, pages 391–408, Berlin, Heidelberg, 2007. Springer-Verlag.

[177] 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.

[178] Maira Sutton. Syria Arrests Razan Ghazzawi and Eleven Other Activists in Renewed Crackdown of Online Dissent, February 2012. `https://www.eff.org/deeplinks/2012/02/twelve-syrian-activists-arrested-amid-renewed-crackdown/`, Last visited: April 15, 2012.

[179] 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.

[180] J.M. Tarascon and M. Armand. Issues and challenges facing rechargeable lithium batteries. *Nature*, 414(6861):359–367, 2001.

[181] D.B. Terry. Caching hints in distributed systems. *IEEE Trans. Softw. Eng.*, 13(1):48–54, 1987.

[182] The Legion of the Bouncy Castle. Bouncy Castle Cryptography Library. `http://www.bouncycastle.org/`, Last visited: March 12, 2012.

[183] 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.

[184] 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.

[185] Sumair Ur Rahman, Urs Hengartner, Usman Ismail, and S. Keshav. Practical security for rural internet kiosks. In *NSDR '08: Proceedings of the second ACM SIGCOMM workshop on Networked systems for developing regions*, pages 13–18, New York, NY, USA, 2008. ACM.

[186] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical report, 2000.

[187] Jennifer Valentino-Devries, Paul Sonne, and Nour Malas. U.S. Firm Acknowledges Syria Uses Its Gear to Block Web, October 2011. `http://online.wsj.com/article/SB10001424052970203687504577001911398596328.html`, Last visited: April 19, 2012.

[188] D.K. Vassilakis and V. Vassalos. Modelling real p2p networks: The effect of altruism. In *Peer-to-Peer Computing, 2007. P2P 2007. Seventh IEEE International Conference on*, pages 19 –26, sept. 2007.

[189] Sonia Verma. Ottawa demands release of Irans 'blogfather', December 2010. `http://www.theglobeandmail.com/news/world/africa-mideast/canadian-iranian-blogger-sentenced-to-19-years-in-prison/article1729730/`, Last visited: April 15, 2012.

[190] Marc Waldman and David Mazières. Tangler: a censorship-resistant publishing system based on document entanglements. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, CCS '01, pages 126–135, New York, NY, USA, 2001. ACM.

[191] Marc Waldman, Aviel D. Rubin, and Lorrie Faith Cranor. Publius: a robust, tamper-evident, censorship-resistant web publishing system. In *Proceedings of the 9th conference on USENIX Security Symposium - Volume 9*, pages 5–5, Berkeley, CA, USA, 2000. USENIX Association.

[192] W. Wang, V. Srinivasan, and M. Motani. Adaptive contact probing mechanisms for delay tolerant applications. In *MobiCom '07: Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 230–241, New York, NY, USA, 2007. ACM.

[193] Y. Wang, S. Jain, M. Martonosi, and K. Fall. Erasure-coding based routing for opportunistic networks. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 229–236, New York, NY, USA, 2005. ACM.

[194] D.L. Wheeler. Working around the state: Internet use and political identity in the arab world. *Routledge Handbook of Internet Politics. London: Routledge*, pages 305–320, 2009.

[195] Wi-Fi Alliance. Wi-Fi Direct, 2012. `http://www.wi-fi.org/discover-and-learn/wi-fi-direct`, Last visited: April 1, 2012.

[196] Mark Wood. SMS bulk messaging, the problem and the solution, March 2009. `http://www.docstoc.com/docs/44720902/SMS-Mass-Messaging-the-problem-and-the-solution`, Last visited: April 1, 2012.

[197] Eiko Yoneki, Pan Hui, ShuYan Chan, and Jon Crowcroft. A socio-aware overlay for publish/subscribe communication in delay tolerant networks. In *MSWiM '07: Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems*, pages 225–234, New York, NY, USA, 2007. ACM.

[198] Jillian C. York. Moroccan Activist's Arrest Signals Crackdown on Speech, February 2012. https://www.eff.org/deeplinks/2012/02/moroccan-activists-arrest-signals-crackdown-speech/, Last visited: April 15, 2012.

[199] M.J. Yuan. *Enterprise J2ME: developing mobile Java applications*. Prentice Hall PTR, 2004.

[200] G.V. Záruba and D. Levine. Accelerated Neighbor Discovery in Bluetooth Based Personal Area Networks. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications-Volume 4 table of contents*, pages 1767–1773. CSREA Press, 2002.

[201] 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 *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 263–268, New York, NY, USA, 2006. ACM.

[202] Kim Zetter. BlackBerry Spyware Wasn't Ready for Prime Time, 2009. http://www.wired.com/threatlevel/2009/07/blackberry-spyware/, Last visited: February 12, 2012.

[203] Chao Zhang, Prithula Dhungel, Di Wu, and Keith W. Ross. Unraveling the bittorrent ecosystem. *IEEE Trans. Parallel Distrib. Syst.*, 22(7):1164–1177, July 2011.

[204] Xiaolan Zhang, Jim Kurose, Brian Neil Levine, Don Towsley, and Honggang Zhang. Study of a bus-based disruption-tolerant network: mobility modeling and impact on routing. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, MobiCom '07, pages 195–206, New York, NY, USA, 2007. ACM.

[205] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 187–198, New York, NY, USA, 2004. ACM.

[206] Wenrui Zhao and Mostafa H. Ammar. Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks. In *FTDCS '03: Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'03)*, page 308, Washington, DC, USA, 2003. IEEE Computer Society.

[207] G. Zhong, I. Goldberg, and U. Hengartner. Louis, Lester and Pierre: Three Protocols for Location Privacy. *Lecture Notes in Computer Science*, 4776:62, 2007.

[208] Zhenyun Zhuang, Tae-Young Chang, Raghupathy Sivakumar, and Aravind Velayutham. A 3: application-aware acceleration for wireless data networks. In *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*, pages 194–205, New York, NY, USA, 2006. ACM.

[209] B. Zorn. *Comparative performance evaluation of garbage collection algorithms*. PhD thesis, University of California, Berkeley, 1989.