APPLICATION ACCELERATION FOR WIRELESS AND MOBILE DATA NETWORKS

A Thesis
Presented to
The Academic Faculty

by

Zhenyun Zhuang

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Computing

Georgia Institute of Technology
December 2010

APPLICATION ACCELERATION FOR WIRELESS AND MOBILE DATA NETWORKS

Approved by:

Dr. Raghupathy Sivakumar,
Committee Chair
Department of ECE
Georgia Institute of Technology

Dr. Raghupathy Sivakumar, Advisor
Department of ECE
Georgia Institute of Technology

Dr. Mostafa H. Ammar
College of Computing
Georgia Institute of Technology

Dr. Umakishore Ramachandran
College of Computing
Georgia Institute of Technology

Dr. Chuanyi Ji
Department of ECE
Georgia Institute of Technology

Dr. Faramarz Fekri
Department of ECE
Georgia Institute of Technology

Date Approved: August 25th, 2010

To my family.

ACKNOWLEDGEMENTS

I am grateful to the help of many individuals, without whom this dissertation would not have been possible.

First and foremost, I would like to express my sincere gratitude to my PhD advisor, Professor. Raghupathy Sivakumar. He instructed me the methodologies and approaches both for performing academic research and presenting research works. He also gave me freedom in making my own decisions and supported me along the way. During my PhD study years, I have always been inspired by his strive for excellence, passion for research, and his innovative mind.

I am also indebted to Dr. Sivakumar for the precious opportunity of conducting my research in GNAN group, an inspiring and productive research group he created and manage. Thanks to his thoughtful guidance and management, our research group is a great environment for developing skills, exchanging ideas, and eventually generating high-quality research works. I remember the numerous insightful paper discussions, the idea-stimulating brainstorming sessions, and the constructive presentation dry-runs. It is my great honor to be a member of the GNAN group. This is an unique experience that I will always be proud of and benefit from for my entire life.

I would like to thank my dissertation committee members: Dr. Mostafa H. Ammar, Dr. Chuanyi Ji, Dr. Faramarz Fekri, and Dr. Umakishore Ramachandran for serving in my thesis proposal and defense committee. I am grateful for the time they spent in reading my thesis and their valuable advices and comments, which helped me improve the quality of this thesis.

I extend my thanks to my current and past labmates in the GNAN research group. Special thanks to Tae-Young Chang, Yujie Zhu, Ramanuja Vedantham, Sandeep Kakumanu, Aravind Velayutham, Sriram Lakshmanan, Cheng-Lin Tsao, Shruti Sanadhya, Karthikeyan Sundaresan, and Dr. Yeonsik Jeong for their collaborations and suggestions in my thesis

research. Finally I would like to thank my family: my wife, my son, my parents, my siblings and my nieces, for always being there for me. Their encouragement and advices helped me through the good times and bad times. Their care and support will always accompany me through my adventures.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This work studies application acceleration for wireless and mobile data networks. The problem of accelerating application can be addressed along multiple dimensions. The first dimension is *advanced network protocol design*, i.e., optimizing underlying network protocols, particulary transport layer protocol and link layer protocol.

Despite advanced network protocol design, in this work we observe that certain application behaviors can fundamentally limit the performance achievable when operating over wireless and mobile data networks. The performance difference is caused by the complex application behaviors of these non-FTP applications. Explicitly dealing with application behaviors can improve application performance for new environments. Along this *overcoming application behavior* dimension, we accelerate applications by studying specific types of applications including Client-server, Peer-to-peer and Location-based applications. In exploring along this dimension, we identify a set of application behaviors that significantly affect application performance. To accommodate these application behaviors, we firstly extract general design principles that can apply to any applications whenever possible. These design principles can also be integrated into new application designs. We also consider specific applications by applying these design principles and build prototypes to demonstrate the effectiveness of the solutions.

In the context of application acceleration, even though all the challenges belong to the two aforementioned dimensions of *advanced network protocol design* and *overcoming application behavior* are addressed, application performance can still be limited by the underlying network capability, particularly physical bandwidth. In this work, we study the possibility of speeding up data delivery by *eliminating traffic redundancy* present in application traffics. Specifically, we first study the traffic redundancy along multiple dimensions using traces obtained from multiple real wireless network deployments. Based on the insights obtained from the analysis, we propose *Wireless Memory (WM)*, a two-ended AP-client

solution to effectively exploit traffic redundancy in wireless and mobile environments.

Application acceleration can be achieved along two other dimensions: network provisioning and quality of service (QoS). Network provisioning allocates network resources such as physical bandwidth or wireless spectrum, while QoS provides different priority to different applications, users, or data flows. These two dimensions have their respective limitations in the context of application acceleration.

In this work, we focus on the two dimensions of *overcoming application behavior* and *Eliminating traffic redundancy* to improve application performance. The contribution of this work is as follows. First, we perform experimental analysis of the new and orthogonal dimensions of application behavior and traffic redundancy to establish them as both viable and necessary optimization avenues for performance in wireless data networks beyond conventional protocol optimization. Second, we extract generalized design principles and develop application-aware acceleration and wireless memory algorithms to exploit the new optimization dimensions. Third, we design and develop real-world solutions that manifest the principles and algorithms to lend a strong systems-oriented focus to the solutions and to serve as credible evaluation platforms.

CHAPTER I

INTRODUCTION

This work studies application acceleration for wireless and mobile data networks. The problem of accelerating application can be addressed along multiple dimensions. The first dimension is *advanced network protocol design*. As applications have to rely on underlying network protocols (e.g., transport layer) for data delivery, the straightforward approach for application acceleration is to optimize underlying network protocols, particulary transport layer protocol and link layer protocol. Compared to their wired and fixed counterparts, wireless and mobile environments typically have unique characteristics such as mobility, high loss rate, large delay and low bandwidth. Because of this, conventional network protocols such as TCP do not perform effectively in wireless and mobile environments. To deal with such characteristics, a significant amount of research has been done toward the development of better lower layer network protocols including transport layer protocols [44,66,70,108] and link layer protocols [36, 53, 119] over the past several decades. Such protocols, and several more, have novel and unique design components that are indeed important for tackling the unique characteristics of wireless environments.

Despite advanced network protocol design, in this work we observe that certain application behaviors can fundamentally limit the performance achievable when operating over wireless and mobile data networks. Specifically, though advanced transport protocols can significantly improve the throughput of FTP, certain non-FTP applications do not see much improvement even when advanced transport protocols are applied. The performance difference is caused by the complex application behaviors of these non-FTP applications. Such behaviors stem from the design of the applications, which is typically tailored for operation in either conventional fixed and wired environments or certain usage scenarios. Explicitly dealing with application behaviors can improve application performance for new environments. Along this *overcoming application behavior* dimension, we accelerate applications by

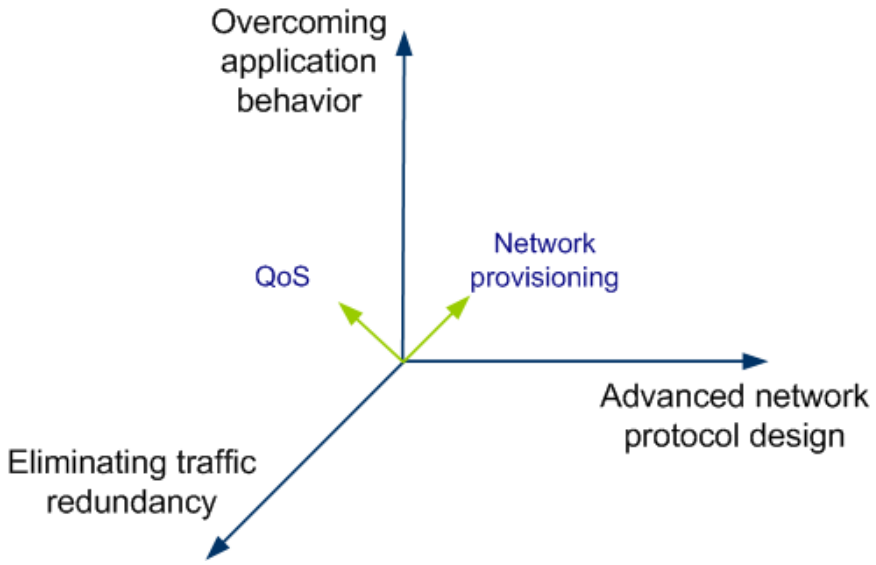


Figure 1: Application acceleration dimensions

studying specific types of applications. First, many conventional applications such as web-based applications are designed for conventional environments, but they suffer performance degradation when moving into wireless and mobile environments. These applications can be largely divided into two categories: Client-server and Peer-to-peer. In addition to conventional applications, there are also mobile-specific applications that specifically designed for mobile environments. One important type of such applications are location-based applications for smartphones. These applications also face challenges that prevent them from working efficiently. Our work optimizes all these three types of applications. In exploring along this dimension, we identify a set of application behaviors that significantly affect application performance. To accommodate these application behaviors, we firstly extract general design principles that can apply to any applications whenever possible. These design principles can also be integrated into new application designs. We also consider specific applications by applying these design principles and build prototypes to demonstrate the effectiveness of the solutions.

In the context of application acceleration, even though all the challenges belong to the two aforementioned dimensions of *advanced network protocol design* and *overcoming application behavior* are addressed, application performance can still be limited by the underlying network capability, particularly physical bandwidth. In other words, even with

perfect designs of protocols across all layers (i.e., from application layer to physical layer), the raw data delivery rate is still constrained by the physical available bandwidth. In this work, we study the possibility of speeding up data delivery by *eliminating traffic redundancy* present in application traffics. Specifically, Several recent studies [51, 87, 105, 111] have shown the presence of considerable amounts of redundancy in Internet traffic content. Such redundancies in content can be explicitly eliminated to improve communication performance. There are various approaches [37, 38, 87, 102, 111] that have been proposed to eliminate such redundancy. Ranging from application-layer to network layer strategies, these works invariably focus on fixed wireline networks. Similar to the above works, we too explore leveraging network traffic redundancy, but exclusively focus on wireless and mobile environments. Unlike wireline networks, wireless and mobile environments exhibit unique challenges and opportunities in the context of redundancy elimination. On one hand, the broadcast nature of wireless communication enables techniques such as packet sniffing to be performed with ease, while on the other hand, mobility and location based channel variances could impose challenges that have to be effectively addressed. In this work, we first study the traffic redundancy along multiple dimensions using traces obtained from multiple real wireless network deployments. Based on the insights obtained from the analysis, we propose *Wireless Memory (WM)*, a two-ended AP-client solution to effectively exploit traffic redundancy in wireless and mobile environments.

Application acceleration can be achieved along two other dimensions: network provisioning and quality of service (QoS). Network provisioning allocates network resources such as physical bandwidth or wireless spectrum, while QoS provides different priority to different applications, users, or data flows. These two dimensions have their respective limitations in the context of application acceleration. Network provisioning may incur substantial cost (e.g., cost of applying new wireless spectrum) and oftentimes is hard to justify. QoS typically provides resource reservation control mechanisms for certain high-priority applications by essentially penalizing other low-priority applications. Two distinctly different philosophies were developed to engineer preferential treatment for packets which require it: IntServ [48] and DiffServ [47].

In this work, we focus on the two dimensions of *overcoming application behavior* and *Eliminating traffic redundancy* to improve application performance, and such focus is primarily because of the comparatively less work available along these dimensions.

1.1 Thesis Contribution and Summary

The contribution of this work is as follows. First, we perform experimental analysis of the new and orthogonal dimensions of application behavior and traffic redundancy to establish them as both viable and necessary optimization avenues for performance in wireless data networks beyond conventional protocol optimization. Second, we extract generalized design principles and develop application-aware acceleration and wireless memory algorithms to exploit the new optimization dimensions. Third, we design and develop real-world solutions that manifest the principles and algorithms to lend a strong systems-oriented focus to the solutions and to serve as credible evaluation platforms.

In summary, our thesis involves the experimental analysis of the new dimensions of application behavior and traffic redundancy for performance optimization in wireless data networks, and design of application-acceleration and wireless memory solutions guided by a strong systems-focus to exploit those dimensions using generalized principles derived from the analysis.

The remainder of the thesis is organized as follows. The next three chapters are along the dimension of overcoming application behavior. Specifically, Chapters 2 and 3 presents the acceleration of client-server applications and peer-to-peer applications, respectively. Chapter 4 focuses on the acceleration of location-based applications. The dimension of eliminating traffic redundancy is explored in Chapter 5. Finally, we conclude the thesis and discuss future work in Chapter 6.

CHAPTER II

ACCELERATING CLIENT-SERVER APPLICATIONS FOR WIRELESS DATA NETWORKS: DESIGN ELEMENTS AND PROTOTYPE IMPLEMENTATION

2.1 Summary

A tremendous amount of research has been done toward improving transport-layer performance over wireless data networks. The improved transport layer protocols are typically application-unaware. In this chapter, we argue that the behavior of applications can and does dominate the actual performance experienced. More importantly, we show that for practical applications, application behavior all but completely negates any improvements achievable through better transport layer protocols. In this context, we motivate an application-aware, but application transparent, solution suite called A^3 (application-aware acceleration) that uses a set of design principles realized in an application specific fashion to overcome the typical behavioral problems of applications. We demonstrate the performance of A^3 through both emulations using realistic application traffic traces and implementations using the NetFilter utility.

2.2 Introduction

A significant amount of research has been done toward the development of better transport layer protocols that can alleviate the problems Transmission Control Protocol (TCP) exhibits in wireless environments [44, 66, 70, 108]. Such protocols, and several more, have novel and unique design components that are indeed important for tackling the unique characteristics of wireless environments. However, in this work we ask a somewhat orthogonal question in the very context the above protocols were designed for: *How does the application's behavior impact the performance deliverable to wireless users?*

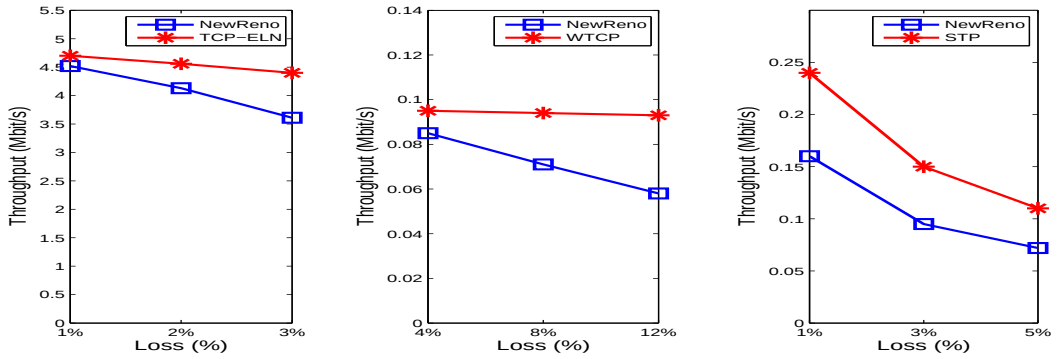
Toward answering this question, we explore the impact of typical wireless characteristics

on the performance experienced by the applications for very popularly used real-world applications including File Transfer Protocol (FTP), the Common Internet File Sharing (CIFS) protocol [4], the Simple Mail Transfer Protocol (SMTP), and the Hyper-Text Transfer Protocol (HTTP). Through our experiments, we arrive at an impactful result: Except for FTP, which has a simple application layer behavior, for all other applications considered, not only is the performance experienced when using vanilla TCP-NewReno much worse than for FTP, but the applications see negligible or no performance enhancements even when they are made to use the wireless-aware protocols.

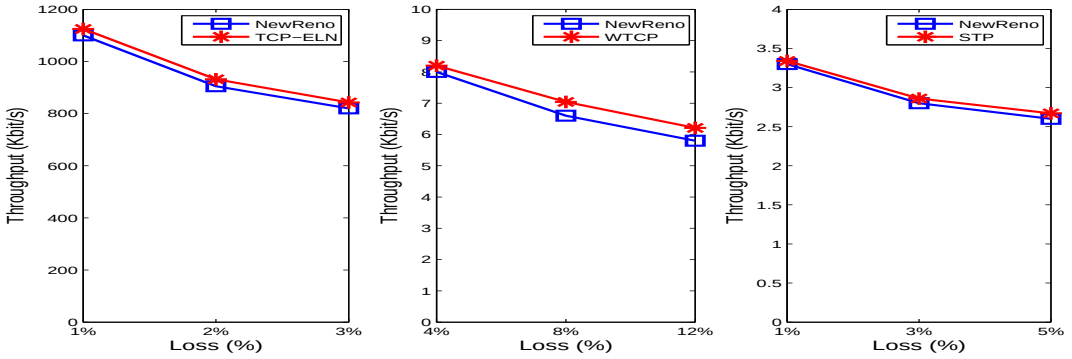
We delve deeper into the above observation and identify several common behavioral characteristics of the applications that fundamentally limit the performance achievable when operating over wireless data networks. Such characteristics stem from the design of the applications, which is typically tailored for operation in substantially higher quality local area network (LANs) environments. Hence, we pose the question: *if application behavior is a major cause for performance degradation as observed through the experiments, what can be done to improve the end-user application performance?*

In answering the above question, we present a new solution called *Application-Aware Acceleration* (A^3, pronounced as "A-cube"), which is a middleware that offsets the typical behavioral problems of real-life applications through an effective set of principles and design elements. A^3's design has five underlying design principles including transaction prediction, prioritized fetching, redundant and aggressive retransmissions, application aware encoding, and infinite buffering. The design principles are derived explicitly with the goal of addressing the aforementioned application layer behavioral problems. We present A^3 as a platform solution requiring entities at both ends of the end-to-end communication, but also describe a variation of A^3 called $A^3\bullet$ (pronounced as "A-cube dot"), which is a point solution but is not as effective as A^3. One of the keystone aspects of the A^3 design is that it is *application-aware, but application transparent.*

The rest of the chapter is organized as follows: Section 2.3 presents the motivation results for A^3. Section 2.4 presents the key design elements underlying the A^3 solution. Section 2.5 describes the realization of A^3 for specific applications. Section 2.6 evaluates



(a) FTP (WLAN, WWAN, and SAT)



(b) CIFS (WLAN, WWAN, and SAT)

A^3 and presents a proof-of-concept prototype of A^3 using the NetFilter utility. Section 2.8 discusses related works, and Section 2.7 concludes the chapter.

2.3 Motivation

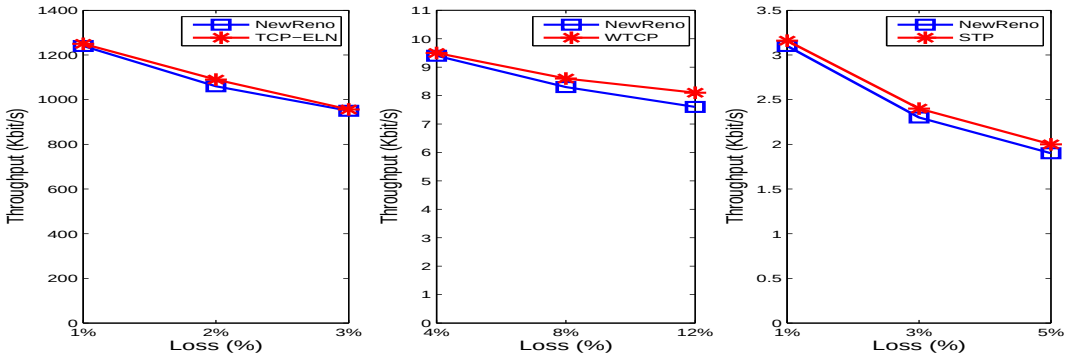
The focus of this work is entirely on applications that require reliable and in-sequence packets delivery. In other words, we consider only applications that are traditionally developed with the assumption of using the TCP transport layer protocol.

2.3.1 Evaluation Model

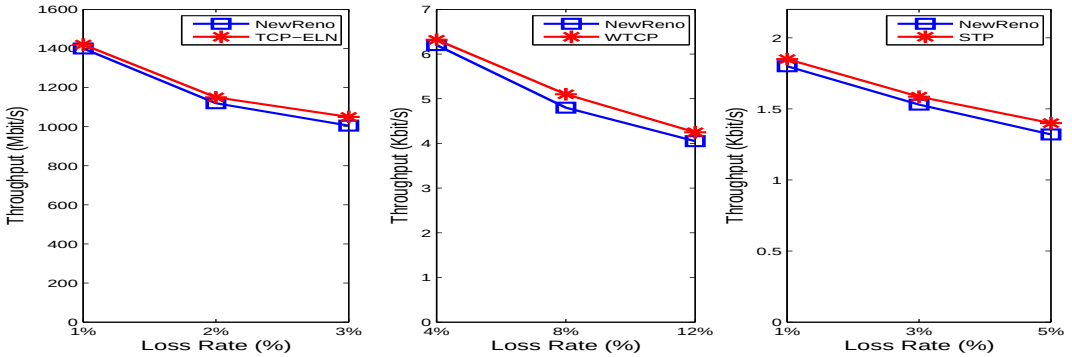
We now briefly present the setting and methodology employed for the results presented in the rest of the section.

2.3.1.1 Applications

For the results presented in this section, we consider four different applications: FTP, CIFS, SMTP and HTTP.



(c) SMTP (WLAN, WWAN, and SAT)



(d) HTTP (WLAN, WWAN, and SAT)

Figure 2: Impact of Wireless Environment Characteristics on Application Throughput

- **CIFS** Common Internet File System is a platform-independent network protocol used for sharing files, printers, and other communication abstractions between computers. While originally developed by Microsoft, CIFS is currently an open technology that is used for all Windows workgroup file sharing, NT printing, and the Linux Samba server[1].

- **SMTP** Simple Mail Transfer Protocol is used for the exchange of emails either between mail servers, or between a client and its server. Most email systems that use the Internet for communication use SMTP.

- **HTTP** HyperText Transfer Protocol is the underlying protocol used by the World Wide Web (WWW).

[1]Samba uses SMB, on which CIFS is based.

2.3.1.2 Traffic generator

We use IxChariot [73] to generate accurate application specific traffic patterns. IxChariot is a commercial tool for emulating most real-world applications. It is comprised of the IxChariot console (for control), performance end-points (for traffic generation and reception), and IxProfile (for characterizing performance).

2.3.1.3 Testbed

We use a combination of a real test-bed and emulation to construct the test-bed for the results presented in the section. Since IxChariot is a software tool that generates actual application traffic, it is hosted on the sender and the receiving machines as shown in Figure 10(b). The path from the sender to the receiver goes through a node running the Network Simulator (NS2) [114] in emulation mode. The network emulator is configured to represent desired topologies including the different types of wireless technologies. More information on the test-bed is presented in Section 2.6.

2.3.1.4 Transport protocols

Since we consider wireless LANs (WLAN), wireless WANs (WWAN), and wireless satellite area networks (SAT), we use transport layer protocols proposed in related literature for each of these environments. Specifically, we use TCP-ELN (NewReno with Explicit Loss Notification) [44], WTCP (Wide-area Wireless TCP) [108], and STP (Satellite Transport Protocol) [66] as enhanced transport protocols for WLANs, WWANs, and SATs respectively.

2.3.1.5 Parameters

We use average RTT values of 5 ms, 200 ms, and 1000 ms, average loss rates of 1%, 8%, and 3%, and average bandwidths of 5 Mbps, 0.1 Mbps, and 1 Mbps for WLANs, WWANs, and SATs, respectively. We simulate wireless channels by introducing various link parameters to packet level traffic with NS2 emulation. The default Ethernet LAN MAC protocol is used. The purpose for such simplified wireless setup is to examine the impact of application behaviors better by isolating the effect of complicated wireless MAC protocols. We use

application-perceived throughput as the key metric of interest. Each data point is taken as an average of 10 different experimental runs.

2.3.2 Quantitative Analysis

Figure 2(a) presents the performance results for FTP under varying loss conditions in WLANs, WWANs, and SAT environments. The tailored protocols uniformly show considerable performance improvements. The results illustrate that the design of the enhancement protocols such as TCP-ELN, WTCP, and STP, is sufficient enough to deliver considerable improvements in performance for wireless data networks, when using FTP as the application. In the rest of the section, we discuss the impact of using such protocols for other applications such as CIFS, SMTP, and HTTP.

Figures 2(b)-(d) show the performance experienced by CIFS, SMTP, and HTTP, respectively, under varying loss conditions for the different wireless environments. It can be observed that *the performance improvements demonstrated by the enhancement protocols for FTP do not carry over to these three applications*. It also can be observed that the maximum performance improvement delivered by the enhancement protocols is less than 5 % across all scenarios.

While the trend evident from the results discussed above is that the enhanced wireless transport protocols do not provide performance improvements for three very popularly used applications, we argue in the rest of the section that this is not due to any fundamental limitations of the transport protocols themselves, but due to the specifics of the behavior of the three applications under consideration.

2.3.3 Impact of Application Behavior

We now explain the lack of performance improvements when using enhanced wireless transport protocols with applications such as CIFS, SMTP, and HTTP. We use the conceptual application traffic pattern for the three applications in Figure 3 for most of our reasonings.

2.3.3.1 Thin session control messages

All three applications, as observed in Figure 3, use *thin session control message exchanges* before the actual data transfer occurs, and *thin request messages* during the actual data transfer phase as well. We use the term "thin" to refer to the fact that such messages are almost always contained in a single packet of MSS (maximum segment size).

The observation above has two key consequences:

• When a loss occurs to a thin message, an entire round trip time (RTT) is taken to recover from such a loss. When the round-trip time is large like in WWANs and SATs, this can result in considerably inflating the overall transaction time for the applications. Note that a loss during the data phase will not have such an adverse impact, as the recovery from that loss can be multiplexed with other new data transmissions whereas for thin message losses, no other traffic can be sent anyway.

• Most protocols, including TCP, rely on the arrival of out-of-order packets to infer packet losses and hence trigger loss recovery. In the case of thin messages, since there are no packets following the lost message, the only means for loss detection is the expiry of the retransmission timer. Retransmission timers typically have coarse minimum values to keep overheads low. TCP, for example, typically uses a minimum Retransmission Time Out (RTO) value of one second.[2]

2.3.3.2 Block-based data fetches

Another characteristic of the applications, especially CIFS and HTTP, is that although the total amount of data to be fetched can be large, the data transfer is performed in blocks, with each block including a "request-response" exchange. CIFS uses its *request-data-block* message to send the block requests, with each request typically requesting only 16 KB to 32 KB of data.

Such a block-based fetching of data has two implications to performance: (i) When the size of the requested data is smaller than the Bandwidth Delay Product (BDP), there is

[2]While newer Linux releases have lower minimum RTO values, they still are in the order of several hundred ms.

a gross underutilization of the available resources. Hence, when the SAT network has a BDP of 128 KB, and CIFS uses a 16 KB request size, the utilization is only 12.5 %. (ii) Independent of the size of each requested data block, one *rtt* is spent in sending the next request once the current requested data arrives. When the *RTT* of the path is large like in WWANs and SATs, this can inflate the overall transaction time and hence lower throughput performance.

2.3.3.3 Flow control bottlenecked operations

Flow control is an important function in communication that helps in preventing the source from overwhelming the receiver. In a mobile/wireless setting, flow control can kick in and prove to be the bottleneck for the connection progress due to two reasons: (i) If the application on the mobile device reads slowly or is temporarily halted for some other reason, the receiver buffer fills up and the source is eventually frozen till the buffer empties. (ii) When there are losses in the network, and the receiver buffer size is of the same order as the BDP (which is typically true), flow control can prevent new data transmissions even when techniques such as fast recovery are used due to unavailability of buffer space at the receiver. With fast recovery, the sender inflates the congestion window to compensate the new ACKs received. However, this inflation may be curbed by the flow control mechanism if there is no buffer space on the receiver side.

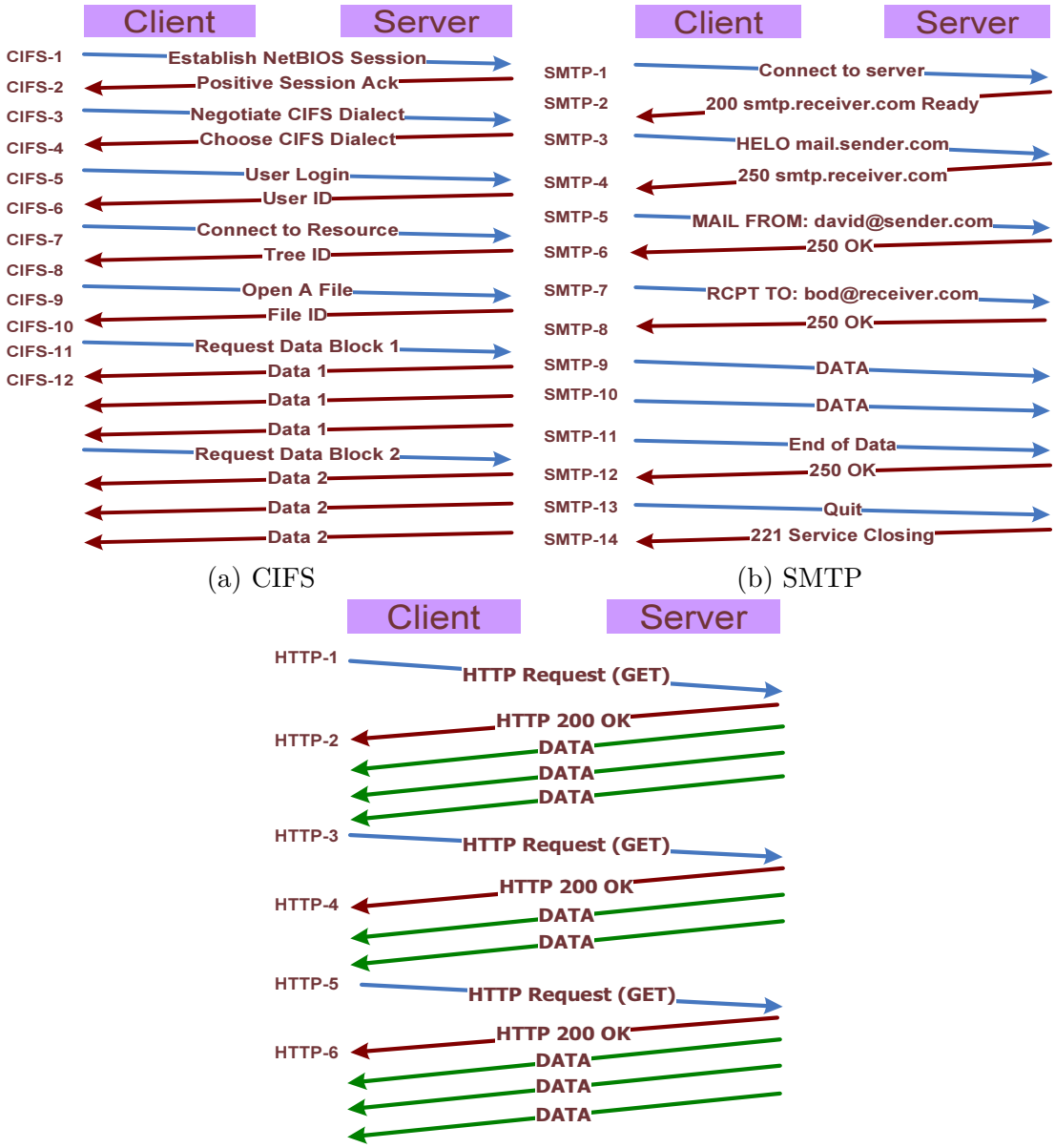
2.3.3.4 Other reasons

While the above discussed reasons are behavioral "acts of commission" by the applications that result in lowered performance, we now discuss two more reasons that can be seen as behavioral "acts of omission". These are techniques that the applications could have used to address conditions in a wireless environment, but do not.

Non-prioritization of data: For all three applications considered, no explicit prioritization of data to be fetched is performed, and hence all the data to be fetched are given equal importance. However, for certain applications prioritizing data in a meaningful fashion can have a profound impact on the performance experienced by the end-system or user. For example, consider the case of HTTP used for browsing on a small-screen PDA. When a

webpage URL request is issued, HTTP fetches all the data for the webpage with equal importance. However, the data corresponding to the *visible* portion of the webpage on the PDA's screen is obviously of more importance and will have a higher impact on the perceived performance by the end-user. Thus, leveraging some means of prioritization techniques can help deliver better performance to the user. With such non-prioritization of data, HTTP suffers performance as defined by the original data size and the low bandwidths of the wireless environment.

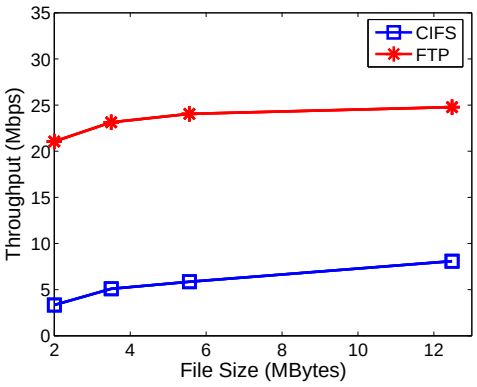
Non-use of data reduction techniques: Finally, another issue is applications not using knowledge specific to their content or behavior to employ effective data reduction techniques. For example, considering the SMTP application, "email vocabulary" of users has evolved over the last couple of decades to be very independent of traditional "writing vocabulary" and "verbal vocabulary" of the users. Hence, it is an interesting question as to whether SMTP can use email vocabulary based techniques to reduce the actual content transferred between SMTP servers, or a SMTP server and a client. Not leveraging such aspects prove to be of more significance in wireless environments where the baseline performance is poor to start with.



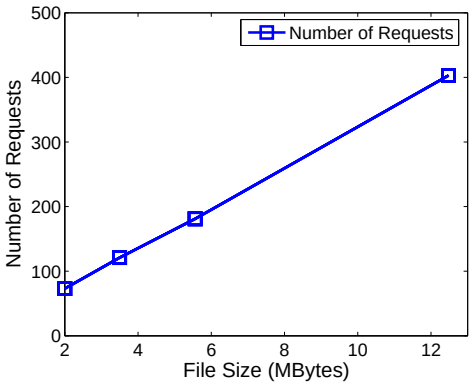
(a) CIFS (b) SMTP

(c) HTTP (Single connection case)

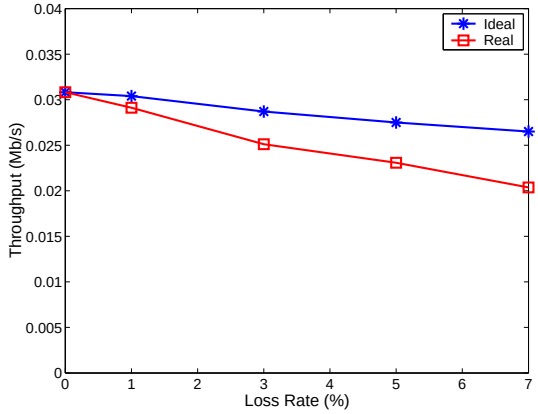
Figure 3: Application Traffic Patterns



(a) Throughput of FTP and CIFS



(b) Number of Requests of CIFS



(c) Throughput of SMTP

Figure 4: Motivation for TP and RAR

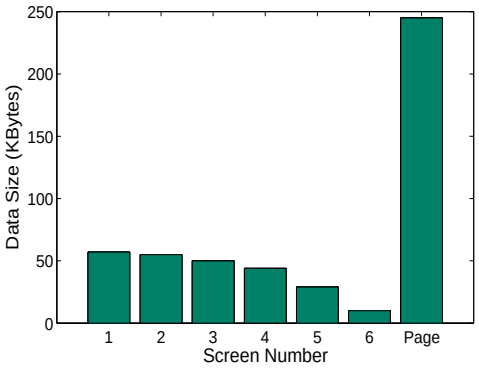
2.4 A^3 Design

Since we have outlined several behavioral problems with applications in Section 2.3, an obvious question to ask is: "*Why not change the applications to address these problems?*" We believe that is indeed one possible solution. Hence, we structure the presentation of the A^3 solution into two distinct components: (i) the key design elements or principles that underlie A^3; and (ii) the actual realization of the design elements for specific applications in the form of an *optimization middleware that is application-aware, but application transparent.* The design elements generically present strategies to improve application behavior and can be used by application developers to improve performance by incorporating changes to the applications directly. In the rest of this section, we outline the design of five principles in the A^3 solution.

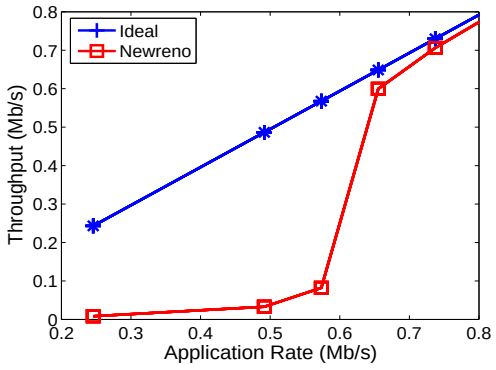
2.4.1 Transaction Prediction (TP)

Transaction prediction (TP) is an approach to deterministically predict future application data requests to the server, and issue them ahead of time. Note that this is different from techniques such as "opportunistic pre-fetching" where content is heuristically fetched to speed up later access but is not guaranteed to be used [3]. In TP, A^3 is fully aware of application semantics and knows exactly what data to fetch and that the data will be used. TP will aid in conditions where the BDP is larger than the default application block fetch size and where the *RTT* is very large. Under both cases, the overall throughput will improve when TP is used. Figure 4(a) shows the throughput performance of CIFS when fetching files of varying sizes in a 100Mbps LAN network. It can be seen that the performance is substantially lower than that of FTP, and this is due to the block based fetching mechanism described in Section 2.3. Figure 4(b) shows the number of transactions it takes CIFS to actually fetch a single file, and it can be observed that the number of transactions increases linearly with file size. Under such conditions, TP will "parallelize" the transactions and hence improve throughput performance. Good examples of applications that will benefit from using TP include CIFS and HTTP for reasons outlined in Section 2.3.

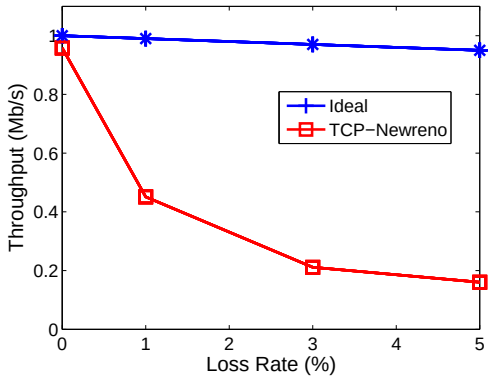
[3] We further discuss on this issue in Section 2.7.



(a) Transfer Size per Screen



(b) Impact of Application Reading Rate



(c) Impact of Loss Increase

Figure 5: Motivation for PF (a) and IB (b, c)

2.4.2 Redundant and Aggressive Retransmissions (RAR)

Redundant and aggressive retransmissions (RAR) is an approach to protect thin session control and data request messages better from losses. The technique involves recognizing thin application messages, and using a combination of packet level redundancy, and aggressive retransmissions to protect such messages. RAR will help address both issues with thin messages identified in Section 2.3. The redundant transmissions reduce the probability of message losses and the aggressive retransmissions that operate on tight *RTT* granularity timeouts reduce the loss recovery time. The key challenges in RAR is to recognize thin messages in an application-aware fashion. Note that *only thin messages require RAR because of reasons outlined in Section 2.3*. Regular data messages should not be subjected to RAR both because their loss recovery can be masked in the overall transaction time by performing the recovery simultaneously with other data packet transmissions, and because

the overheads of performing RAR will become untenable when applied to large volume messages such as the data. Figure 4(c) shows the throughput performance of SMTP under lossy conditions in a WWAN setup. The dramatic effect of a 35 % drop in throughput performance for a loss-rate increase from 0 % to 7 % is much higher than the 15 % drop in performance in the FTP performance for the same corresponding loss-rate increase shown in Section 2.3. Typical applications that can benefit from RAR include CIFS, SMTP, and HTTP.

2.4.3 Prioritized Fetching (PF)

Prioritized fetching (PF) is an approach to prioritize subsets of data to be fetched as being more important than others and to fetch the higher priority data *faster* than the lower priority data. A simple approach to achieve the dual-rate fetching is to use default TCP-like congestion control for the high priority data, but use congestion control like in TCP-LP [78] for low priority data. An important consideration in PF is to devise a strategy to prioritize data intelligently and on the fly. Figure 5(a) shows the average *transfer sizes per screen* as well as the entire web page for the top fifty accessed webpages on the World Wide Web [5]. It can be seen that nearly 80 % of the data (belonging to screens 2 and higher) are not directly impacting response time experienced by the user and hence can be de-prioritized in relation to the data pertaining to the first screen. Note that the results are for a 1024x768 resolution laptop screen, and will in fact be better for smaller screen devices such as PDAs. Good examples of applications that can benefit from PF include HTTP and SMTP.

2.4.4 Infinite Buffering (IB)

Infinite buffering (IB) is an approach that prevents flow control from throttling the progression of a network connection terminating at the mobile wireless device. IB prevents flow control from impacting performance by providing the sender the impression of an *infinite buffer* at the receiver. Secondary storage is used to realize such an infinite buffer, with the main rationale being that *reading from the secondary storage will be faster than fetching it from the sender over the wireless network when there is space created in the actual connection buffer at a later point.* With typical hard-disk data transfer rates today being at

around 250 Mbps [15], the abovementioned rationale is well justified for wireless environments. Note that the trigger for using IB can be both due to application reading slowly or temporarily not reading form the connection buffer, and due to losses on the wireless path. Figures 5(b)-(c) show the throughput performance of SMTP under both conditions. Note that the ideal scenarios correspond to an upper bound of the throughput. [4] It can be observed that for both scenarios, the impact of flow control drastically lowers performance compared to what is achievable. In the rest of the chapter we focus on IB specifically in the context of the more traditional trigger for flow control − application reading bottleneck. Typical applications that can benefit from IB include CIFS, SMTP, and HTTP - essentially, any application that may attempt to transfer more than a BDP worth of data.

2.4.5 Application-aware Encoding (AE)

Table 1: Statistics of 100 Emails Sent by Ten Users

ID	Unique Words	Total Words	Chars / Word	Bits / Email	Simple Coding
1	1,362	6,383	6.22	3,176	665
2	3,554	5,907	7.12	10,984	2,275
3	2,645	12,653	7.08	7,167	1,439
4	4,536	25,481	6.15	12,537	3,095
5	966	4,728	11.46	4,335	469
6	1,205	6,413	5.48	2,811	656
7	798	3,346	4.40	1,178	323
8	1,527	6,836	5.72	3,128	723
9	1,758	9,171	4.91	3,602	989
10	1,402	8,320	7.30	4,859	870

Application-aware encoding (AE) is an approach that uses application specific information to better encode or compress data during communication. Traditional compression tools such as *zip* operate on a given content in isolation without any context for the application corresponding to the content. AE, on the other hand, explicitly uses this contextual information to achieve better performance. Note that AE is not a better compression algorithm. However, it is a better way of identifying data-sets that need to be operated on by a given compression algorithm. Table 1 shows the average email vocabulary characteristics of ten different graduate students based on 100 emails sent by each person during two weeks.

[4] In Figure 5(c), the throughput drop is caused by both flow control and congestion control related mechanisms, and flow control mechanism contributes significantly.

It is interesting to see the following characteristics in the results: (i) the email vocabulary size across the ten people is relatively small – a few thousand words; and (ii) even a simple encoding involving this knowledge will result in every word being encoded with only 10 to 12 bits, which is substantially lower than using 40 to 48 bits required using standard binary encoding. In Section 2.6, we show that such vocabulary based encoding can considerably outperform other standard compression tools such as *zip* as well. Moreover, further benefits can be attained if more sophisticated compression schemes such as Huffman encoding is employed instead of a simple BINARY encoding. Typical applications that can benefit from using AE include SMTP and HTTP.

2.5 A^3 Solution

2.5.1 Deployment Model and Architecture

The A^3 deployment model is shown in Figure 6. Since A^3 is a platform solution, it requires two entities at either end of the communication session that are A^3-aware. At the mobile device, A^3 is a software module that is installed in user space. At the server side, while A^3 can be deployed as a software module on all servers, a more elegant solution would be to deploy a packet processing network appliance that processes all content flowing from the servers to the wide-area network. We assume the latter model for our discussions. However, note that A^3 can be deployed in either fashion as it is purely a software solution.

This deployment model will help in any communication between a server behind the A^3 server and the mobile device running the A^3 module. However, if the mobile device communicates with a non-A^3 enabled server, two options exist: (i) As we discuss later in the chapter, A^3 can be used as a point-solution with lesser effectiveness; or (ii) the A^3 server

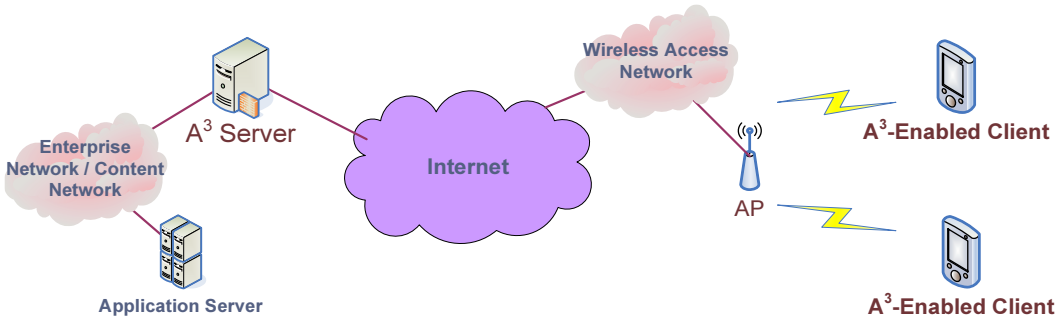


Figure 6: Deployment Model

is brought closer to the mobile device, perhaps within the wireless network provider's access network. In the rest of the chapter, we do not delve into the latter option. However, we do revisit the point-solution mode of operation of A^3.

We present an A^3 implementation that resides in user-space, and uses the NetFilter utility in Linux for the capturing of traffic outgoing and incoming at the mobile device. NetFilter is a Linux specific packet capture tool that has hooks at multiple points in the Linux kernel. The A^3 hooks are registered at the Local-In and Local-Out stages of the chain of hooks in NetFilter. While our discussions are Linux-centric, our discussions can be mapped on the Windows operating system through the use of the Windows Packet Filtering interface, or wrappers such as PktFilter that are built around the interface. Figure 7(a) shows the A^3 deployment on the mobile device using NetFilter.

The A^3 software architecture is shown in Figure 7(b). Since the design elements in A^3 are to a large extent independent of each other, a simple chaining of the elements in an appropriate fashion results in an integrated A^3 architecture. The specific order in which the elements are chained in the A^3 realization is TP, RAR, PF, IB, and AE. While RAR protects the initial session control exchanges and the data requests, it operates on traffic after TP, given that TP can generate new requests for data. PF manipulates the priority with which different requests are served, and IB ensures that data responses are not throttled by flow control. Finally, AE compresses any data outgoing, and decompresses any data incoming.

2.5.2 Application Overviews

Since we describe the actual operations of the mechanisms in A^3 in the context of one of the three applications, we now briefly comment on the specific message types involved in typical transactions by those applications. We then refer to the specific message types when describing the operations of A^3 subsequently.

For simplicity, instead of presenting all message types again, we refer readers back to Figure 3 to observe the message exchanges for the three applications. The labels such as CIFS-x refer to particular message types in CIFS and will be referred to in the A^3 realization descriptions that follow.

CIFS, also sometimes known as Server Message Block (SMB), is a platform independent protocol for file sharing. The typical message exchanges in a CIFS session are as shown in Figure 3(a). Overall, TP manipulates the CIFS-11 message, RAR operates on CIFS-1 through CIFS-11, and IB aids in CIFS-12.

SMTP is Internet's standard host-to-host mail transport protocol and traditionally operates over TCP. The typical message exchanges in an SMTP session are shown in Figure 3(b). Overall, RAR operates on SMTP-1 through SMTP-8, and SMTP-12 through SMTP-14, IB and AE operates on SMTP-9 and SMTP-10.

The HTTP message exchange standard are relatively simple, and typically consist of the messages shown in Figure 3(c). A typical HTTP session consists of multiple objects, as well as the main HTML file, and hence appear as a sequence of overlapping exchanges of the above format. Overall, RAR operates on HTTP-1,2,3,5; and PF and IB operate on HTTP-3,5.

2.5.3 A^3 Realization

In the rest of the section, we take one design element at a time, and walk through the algorithmic details of the element with respect to a single application. Note that A^3 is an application-aware solution, and hence its operations will be application specific. Since we describe each element in isolation, we assume that the element resides between the application and the network. In an actual usage of A^3, the elements will have to be chained as discussed earlier.

2.5.3.1 Transaction Prediction

Figure 8(a) shows the flow chart for the implementation of TP for CIFS at the A^3 client. When A^3 receives a message from the application, it checks to see if the message is CIFS-9, and records state for the file transfer in its File-TP-States data structure. It then passes through the message. If the message was a request, TP checks to see if the request is for a locally cached block, or for a new block. If the latter, it updates the request for more blocks, stores information about the predicted requests generated in the Predicted-Request-States data structure, and forwards the requests.

In the reverse direction, when data comes in from the network, TP checks to see if the data is for a predicted request. If yes, it caches the data in secondary storage and updates its state information, and forwards the data to the application otherwise.

The number of additional blocks to request is an interesting design decision. For file transfer scenarios, TP generates requests asking for the entire file [5]. The file size information can be retrieved from the CIFS-10 message. If the incoming message is for an earlier retrieved block, TP retrieves the block from secondary storage, and provides it to the application.

While CIFS servers accept multiple data requests from the same client simultaneously, it is possible that for some applications, the server might not be willing to accept multiple data requests simultaneously. In such an event, the A^3 server will let only one of the client requests go through to the server at any point in time, and will send the other requests one at a time once the previous requests are served.

2.5.3.2 Redundant and Aggressive Retransmissions

Figure 8(b) shows the flow chart for the implementation of RAR for CIFS. When A^3 receives a message from the application, it checks to see if it is a thin message. The way A^3 performs the check is to see if the message is one of the messages between CIFS-1 and CIFS-11. All such messages are interpreted as thin messages.

If the incoming message is not a thin one, A^3 will let it through as-is. Otherwise, A^3 will create redundant copies of the message, note the information about current time, start retransmission alarm, and send out the copies in a staggering fashion. When a response arrives, A^3 checks the timestamp for the corresponding request, and updates its estimated RTT. A^3 then passes on the message to the application. If the alarm expires for a particular thin message, the message is again subjected to the redundant transmissions. A^3 server is responsible for filtering the successful arrivals of redundant copies of the same message.

The key issues of interest in the RAR implementation are: (i) How many redundant transmissions are performed? Since packet loss rates in wireless data networks rarely exceed

[5] We further discuss on this issue in Section 2.7.

10 %, even a redundancy factor of *two* (two additional copies created) reduces the effective loss-rate to about 0.1 %. Hence, A^3 uses a redundancy factor of two. (ii) How should the redundant messages be staggered? The answer to this question lies in the specific channel characteristics experienced by the mobile device. However, at the same time, the staggered delay should not exceed the round-trip time of the connection, as otherwise the mechanism would lose its significance by unnecessarily delaying the recovery of losses. Hence, A^3 uses a staggering delay of $\frac{RTT}{10}$ between any two copies of the same message. This ensures that within 20 % of the RTT duration, all messages are sent out at the mobile device. (iii) How is the aggressive timeout value determined? Note that while the aggressive timeout mechanism will help under conditions when all copies of a message are lost, the total message overhead by such aggressive loss recovery is negligible when compared to the overall size of data transferred by the application. Hence, A^3 uses a timeout value of the $RTT_{avg} + \alpha$, where α is a small guard constant, and RTT_{avg} is the average RTT observed so far. This simple setting ensures that the timeout values are tight, and at the same time the mechanism adapts to changes in network characteristics.

2.5.3.3 Prioritized Fetching

Figure 9(a) shows the flow chart for the implementation of PF in the context of HTTP. Once again, the key goal in PF for HTTP is to retrieve web objects that are required for the display of the visible portion of the webpage quickly at the expense of the objects on the page that are not visible.

Unlike in the other mechanisms, PF cannot be implemented without some additional interactions with the application itself. Fortunately, browser applications have well defined interfaces for querying state of the browser including the current window focus, scrolling information, etc. Hence, the implementation of PF relies on a separate module called the application state monitor (ASM) that is akin to a browser plug-in to coordinate its operations.

When a message comes in from the application, PF checks to see if the message is a request. If it is not, it is let through. Otherwise, PF checks with the ASM to see if the

requested content are immediately required. ASM classifies the objects requested as being of immediate need (*i.e.*, visible portion of webpage) or as those that are not immediately required. PF then sends out fetch requests immediately for the first category of objects and uses a low-priority fetching mechanism for the remaining objects.

Since A^3 is a platform solution, all PF has to inform the A^3 server is that certain objects are of low priority through A^3-specific piggybacked information. The A^3 server then de-prioritizes the transmission of those objects in preference to those that are of higher priority. Note that the relative prioritization is used not only between the content of a single end-device, but also across end-devices as well to improve overall system performance. Approaches such as TCP-LP [78] are candidates that can be used for the relative prioritization between TCP flows, although A^3 currently uses a simple priority queuing scheme within the same TCP flow at the A^3 server.

Note that while the ASM might classify objects in a particular fashion, changes in the application (e.g. scrolling down) will result in a re-prioritization of the objects accordingly. Hence, the ASM has the capability of gratuitously informing PF about priority changes. Such changes are immediately notified to the A^3 server through appropriate requests.

2.5.3.4 Infinite Buffering

Figure 9(b) shows the flow chart for the implementation of IB in the context of SMTP. IB keeps track of TCP connection status, and monitors all ACKs that are sent out by the TCP connection serving the SMTP application for SMTP-9 and SMTP-10. If the advertised window in the ACK is less than the maximum possible, IB immediately resets the advertised window to the maximum value, and appropriately updates its current knowledge of the connection's buffer occupancy and maximum in-sequence ACK information.

Hence, IB prevents anything less than the maximum buffer size from being advertised. However, when data packets arrive from the network, IB receives the packets and checks to see if the connection buffer can accommodate more packets. If the condition is true, IB delivers the packets to the application directly. If the disk cache is non-empty, which means the connection buffer is full, the incoming packet is directly added to the cache. In this

case, IB generates a proxy ACK back to the server. Then, if the connection buffer has space in it, packets are retrieved from the disk cache and given to the application till the buffer becomes full again. When the connection sends an ACK for a packet already ACKed by IB, IB suppresses the ACK. When the connection state is torn down for the SMTP application, IB resets the state accordingly.

2.5.3.5 Application-aware Encoding

Figure 10 shows the flow-chart for the implementation of AE for SMTP. When AE receives data (SMTP-9) from the SMTP application, it uses its application vocabulary table to compress the data, and marks the message as being compressed and forwards it to the network. The marking is done to inform the A^3 server about the need to perform de-compression. Similarly, when incoming data arrives for the SMTP server, and the data is marked as compressed, AE performs the necessary de-compression.

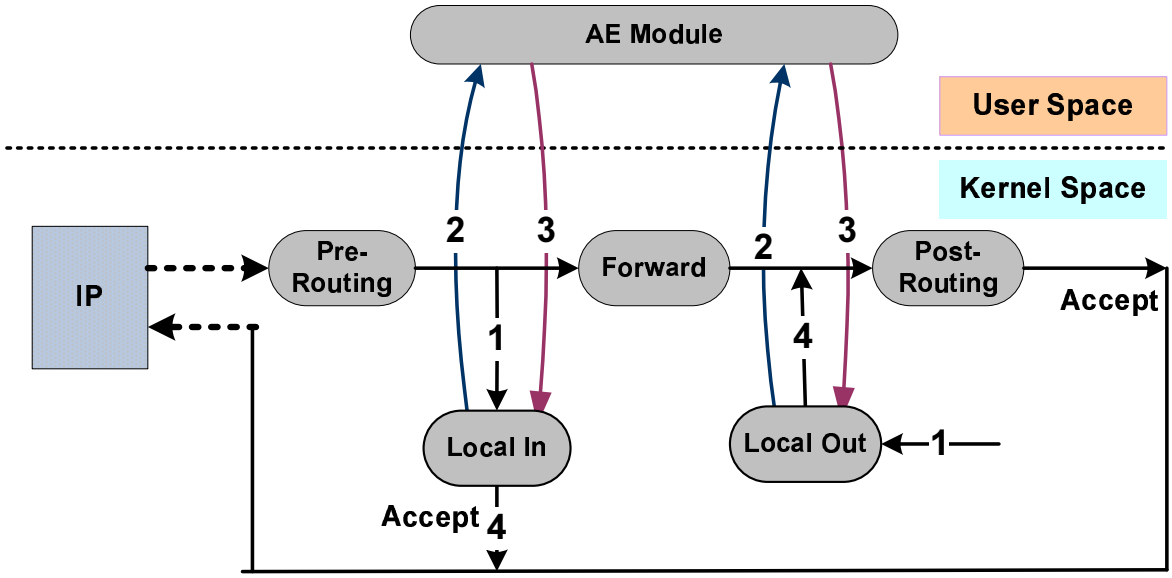
The mechanisms used for the actual creation and manipulation of the vocabulary tables are of importance to AE. In A^3, the SMTP vocabulary tables are created and maintained purely on a user pair-wise basis. Not only are the table created in this fashion, but the data sets over which the vocabulary tables are created is also restricted to this pair-wise model. In other words, if A is the sender and B is the receiver, A uses its earlier emails to B as the data set on which the A-B vocabulary table is created, and then uses this table for encoding. B, having the data set already (since the emails were sent to B), can exactly recreate the table on its side and hence decode any compressed data. This essentially precludes the need for exchanging tables frequently, and also takes advantage of changes in vocabulary sets that might occur based on the recipient. Though the tables are created on both sides implicitly and synchronized in most cases, a backup mechanism used to explicitly synchronize the tables is also needed. The synchronization action is triggered by a mismatch of table hashes on both sides, and the hash is sent along each new email and updated when the table changes.

2.5.4 A^3 Point Solution - $A^3\bullet$

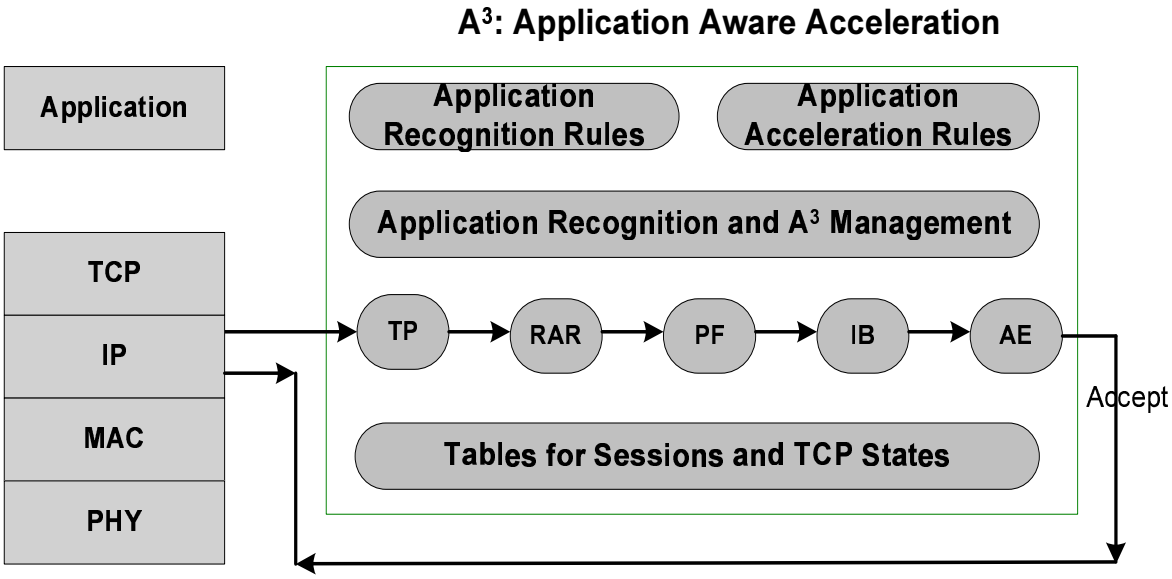
While the A^3 deployment model assumed so far is a platform model requiring participation by A^3 enabled devices at both the client and server ends, in this section we describe how A^3 can be used as a point-solution, albeit with somewhat limited capabilities. We refer to the point-solution version of A^3 as $A^3\bullet$.

Of the five design elements in A^3, the only design element for which the platform model is mandatory is the application-aware encoding mechanism. Since compression or encoding is an end-to-end process, $A^3\bullet$ cannot be used with AE. However, each of the other four principles can be employed with minimal changes in $A^3\bullet$.

TP involves the generation of predictive data requests, and hence can be performed in $A^3\bullet$ as long as the application server can accept multiple simultaneous requests. For CIFS and HTTP, the servers do accept simultaneous requests. IB is purely a flow control avoidance mechanism, and can be realized in $A^3\bullet$. RAR involves redundant transmissions of messages, and hence can be implemented in $A^3\bullet$ as long as application servers are capable of filtering duplicate messages. If the application servers are not capable of doing so (e.g. HTTP servers, which would respond to each request), the redundant transmissions will have to be performed at the granularity of transport layer segments as opposed to application layer messages, since protocols such as TCP provide redundant packet filtering. Finally, PF can be accomplished in $A^3\bullet$ in terms of classifying requests and treating the requests differently. However, the slow fetching of data not required immediately has to be realized through coarser receiver based mechanisms such as delayed requests as opposed to the best possible strategy of slowing down responses as in A^3.

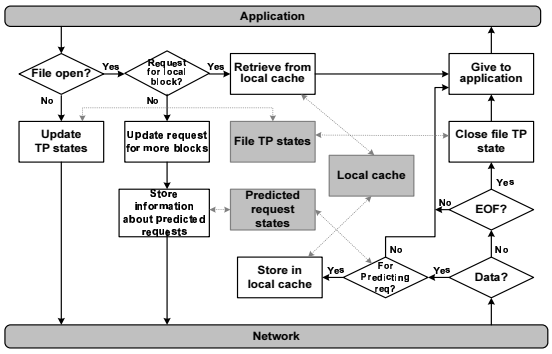


(a) Deployment with Netfilter

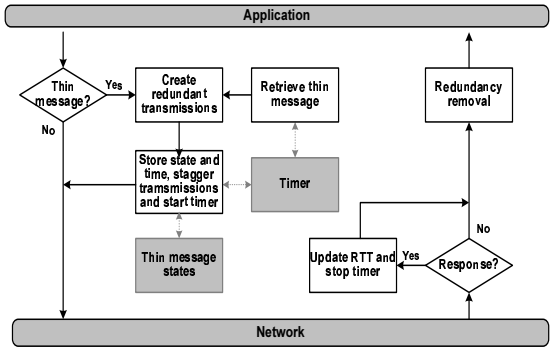


(b) Software Architecture

Figure 7: A^3 Deployment Model with NetFilter and Software Architecture

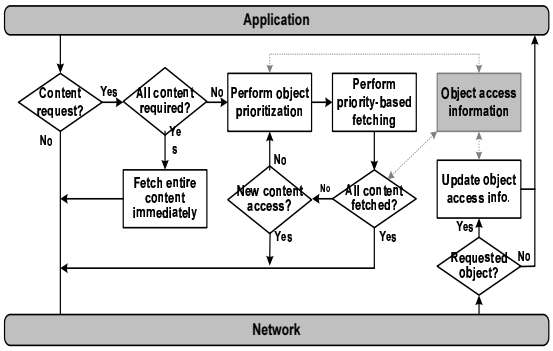


(a) Transaction Prediction

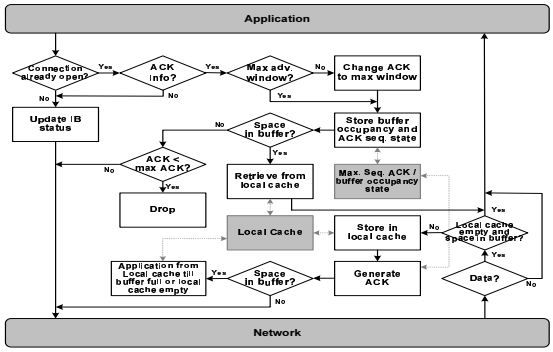


(b) Redundant and Aggressive Retransmissions

Figure 8: TP and RAR (Shaded blocks are storage space and timer, white blocks are operations.)



(a) Prioritized Fetching



(b) Infinite Buffering

Figure 9: PF and IB (Shaded blocks are storage space and timer, white blocks are operations.)

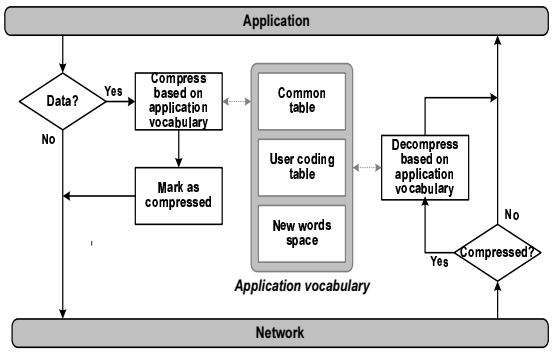


Figure 10: Application-aware Encoding

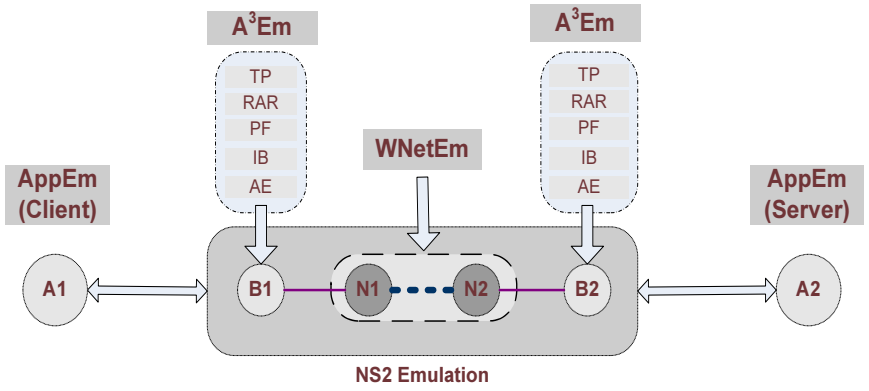


Figure 11: Simulation Network

2.6 Performance Evaluation

In this section we evaluate the performance of A^3. The evaluation is performed with application-specific traffic generators which are modeled based on traffic traces generated by the IxChariot emulator and documented standards for the application protocols. Since each application protocol in the study has various software implementations, and different implementations may differ in certain aspects of protocol standards, we believe such simulations with abstracted traffic generators can help capture the trend of performance enhancement delivered by A^3.

In addition to the emulation, we also build a proof-of-concept prototype of A^3. The prototype implements all five of A^3 design principles and works with the following applications: SCP (Secure Copy), Internet Explorer, Samba, and SendMail. The primary goal of building such a prototype is to prove that the proposed A^3 architecture does indeed work with real applications. We also use the prototype to obtain system level insights into A^3 implementation.

2.6.1 Setup

• *Emulation:* The experimental setup for the emulation is shown in Figure 11. The setup consists of three desktop machines running the Fedora Core 4 operating system with the Linux 2.6 kernel. All the machines are connected using 100 Mbps LAN.

An application-emulator (AppEm) module runs on both the two end machines. The AppEm module is a custom-built user-level module that generates traffic patterns and

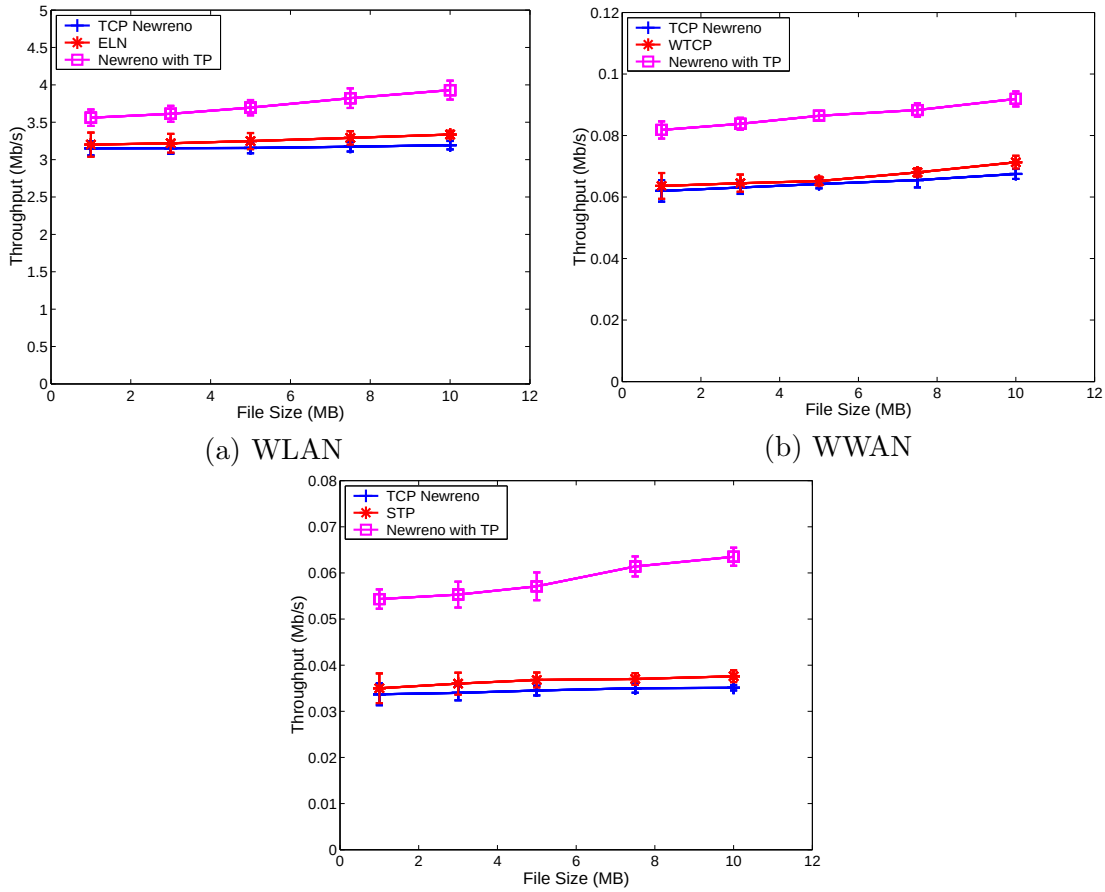


Figure 12: Simulation Results of Transaction Prediction (CIFS)

content for three different application protocols: CIFS, SMTP, and HTTP. The AppEm module also generates traffic content based on both real-life input data-sets (for email and Web content) and random data-sets (File transfer)[6]. The traffic patterns shown in Figure 3 are representative of the traffic patterns generated by AppEm.

The system connecting the two end-systems runs the emulators for both A^3 (A^3-Em) and the wireless network (WNetEm). Both emulators are implemented within the framework of the *ns2* simulator, and ns2 is running in the emulation mode. Running ns2 in its emulation mode allows for the capture and processing of live network traffic. The emulator object in ns2 taps directly into the device driver of the interface cards to capture and inject real packets into the network. All five A^3 mechanisms are implemented in the A^3-Em module,

[6]While the IxChariot emulator can generate representative traffic traces, it does not allow for specific data sets to be used for the content, and hence the need for the custom built emulator.

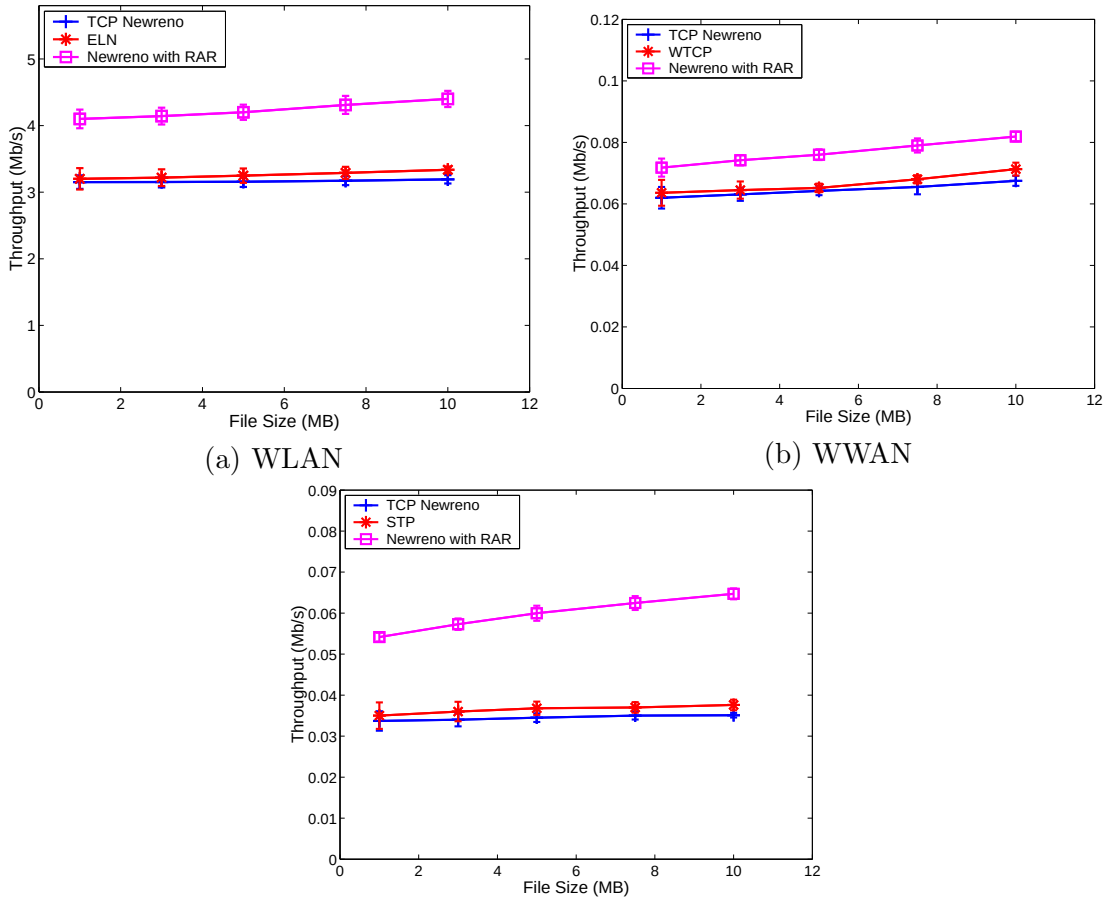


Figure 13: Simulation Results of Redundant and Aggressive Retransmissions (CIFS)

and each mechanism can be enabled either independently or in tandem with the other mechanisms. The WNetEm module is used for emulating different wireless network links representing the WLAN, WWAN, and SAT environments. The specific characteristics used to represent wireless network environments are the same as those presented in Section 2.3.

The primary metrics monitored are throughput, response time (for HTTP) and confidence intervals for the throughput and response time. Each data point is the average of 20 simulation runs and in addition we show the 90 % confidence intervals. The results of the evaluation study are presented in two stages. We first present the results of the performance evaluation of A^3 principles in isolation. Then, we discuss the combined performance improvements delivered by A^3.

• *Proof-of-concept Prototype:* The prototype runs on a testbed consisting of five PCs connected in a linear topology. The first four PCs run Fedora Core 5 with 2.6.15 Linux

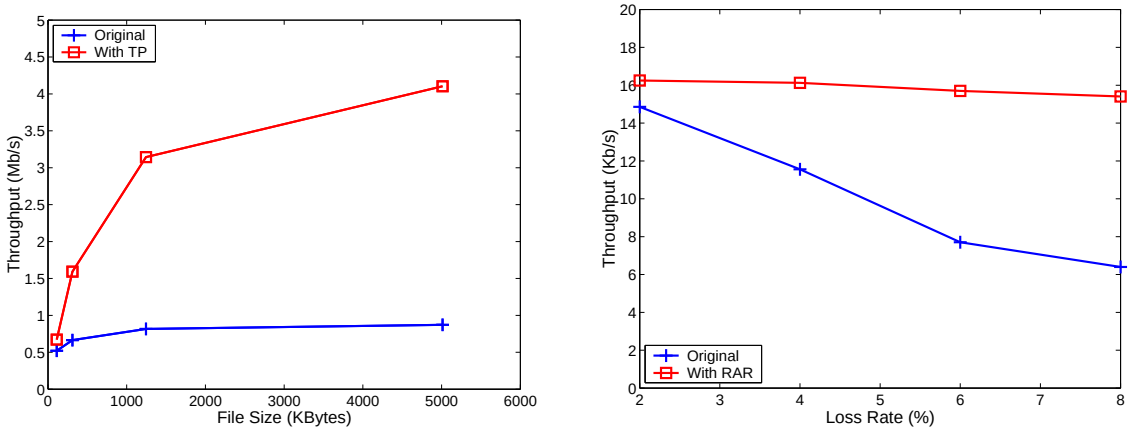
kernel. The fifth PC has dual OS of both Fedora Core 5 and Windows 2000. All machines are equipped with 1 GHz CPU and 256 MB memory. The implementation utilizes NetFilter [21] Utility for Linux platform. The first PC works as the application server for the SMTP (Sendmail server), CIFS (Samba server) and SCP (SCP server). The A^3 server module and client module are installed on the second and fourth PCs, respectively. The third PC works as a WAN emulator, and the fifth PC has the email client, sambaclient, SCP client and Internet Explorer running on it.

The prototype implementation makes use of two netfilter libraries: libnfnetlink (version 0.0.16) and libnetfilter_queue (version 0.0.12) [21]. The registered hook points that we use are NF_IP_FORWARD. For all the hooks, the registered target is the NF_QUEUE, which queues packets and allows user-space processing. After appropriate processing, the queued packets will be passed, dropped, or altered by the modules.

2.6.2 Transaction Prediction

We use CIFS as the application traffic for evaluating the performance of Transaction Prediction. The results of the TP evaluation are shown in Figure 12. The x-axis of each graph shows the size of the transferred file in MBytes and the y-axis the application throughput in Mbps. The results show several trends: (i) Using wireless-aware transport layer protocols (such as ELN, WTCP, and STP), the increase in throughput is very negligible. This trend is consistent with the results in Section 2.3. (ii) Transaction Prediction improves CIFS application throughput significantly. In the SAT network, for instance, TP improves CIFS throughput by more than 80 % when transferring a 10 MByte file. (iii) The improvement achieved by TP increases with increase in file size. This is because TP is able to reduce more the number of request-response interactions with increasing file size. (iv) TP achieves the highest improvement in SAT network.

• *Proof-of-concept Results:* The prototype works with a Smbclient and a Samba server. The scenario considered in the prototype is that of the Smbclient requesting files of various sizes from the Samba server. The implementations for CIFS are working with SMB protocols running directly above TCP (i.e. by connecting to port 445 of Samba servers) instead of



(a) TP (Samba Server) (b) RAR (SendMail)

Figure 14: Prototype Results of TP and RAR

over NetBIOS sessions. The Samba server version is 3.0.23, and smbclient version is 2.0.7.

One of the non-trivial issues faced while implementing the prototype is the TCP sequence manipulation. The issue is caused by the TP acceleration requests generated by A^3. SMB sessions use TCP to request/send data, thus the Samba server is always expecting TCP packets with correct TCP sequence numbers. The acceleration request have to predict not only the block offset for SMB sessions, but also the TCP sequence numbers, failing which the Samba server would see a TCP packet with an incorrect TCP sequence number and behave unexpectedly. The prototype implementation addresses this problem by also keeping track of TCP state and using the appropriate TCP sequence numbers. This is an indication of the application-awareness of A^3 potentially needing to be extended to include transport layer awareness as well. Since some of the principles in A^3 are directly transport layer dependent (e.g. infinite buffering) we believe that this extension still falls within the scope of A^3.

The proof-of-concept results are shown in Figure 14(a). TP helps deliver more improvement for larger files, and the throughput improvement achieved when requesting a 5 MBytes file is up to 500%.

2.6.3 Redundant and Aggressive Retransmissions

We evaluate the effectiveness of RAR using the CIFS application protocol. The results of the RAR evaluation is presented in Figure 13. The x-axis in the graphs is the requested

file size in MB and the y-axis is the CIFS application throughput in Mbps. We observe that RAR delivers better performance when compared to both TCP-NewReno and the tailored transport protocols, delivering up to 80% improvement in throughput performance for SATs. RAR is able to reduce the chances of experiencing a timeout when a wireless packet loss occurs. The reduction of TCP timeouts leads to better performance using RAR.

• *Proof-of-concept Results:* The prototype implementation of RAR includes components for performing the following functions: recognizing session control messages, retransmission control, and redundancy removal. On the sender side, the retransmission control component maintains current RTT values, sets timers, and retransmits the possibly lost messages when timers expire. The transmission of the redundant messages is done using raw sockets. On the receiver side, the redundancy removal component identifies redundant messages when the retransmission is a false alert, *i.e.*, the original message being retransmitted was not lost, but the RAR aggressively performs retransmission.

The prototype is built with a SendMail server and an email client. An email of 5.2 KB is sent to SendMail server over the network of 200 ms RTT, 100 Kbps bandwidth and varying loss rates. Throughput with and without RAR is shown in Figure 14(b). We observe a 250% throughput improvement when loss rate is 8%. More interestingly, the throughput of RAR is not affected much by the loss rate since RAR effectively hides the losses.

2.6.4 Infinite Buffering

The effectiveness of IB is evaluated using CIFS traffic, and the results are shown in Figure 15. The x-axis is requested file size in MBytes and the y-axis are the application throughput in Mbps. We can see that: (i) Transferring larger data size with IB achieves higher throughput. This is because that IB helps most during the actual data transfer phase, and will not help when the amount of data to be transferred is less than a few times the BDP of the network. (ii) IB performs much better in a SAT network than the other two networks, delivering almost a 400 % improvement in performance. Again, the results are as expected because IB's benefits are higher when the BDP of the network is higher.

• *Proof-of-concept Results:* We choose an SCP implementation to build the prototype of

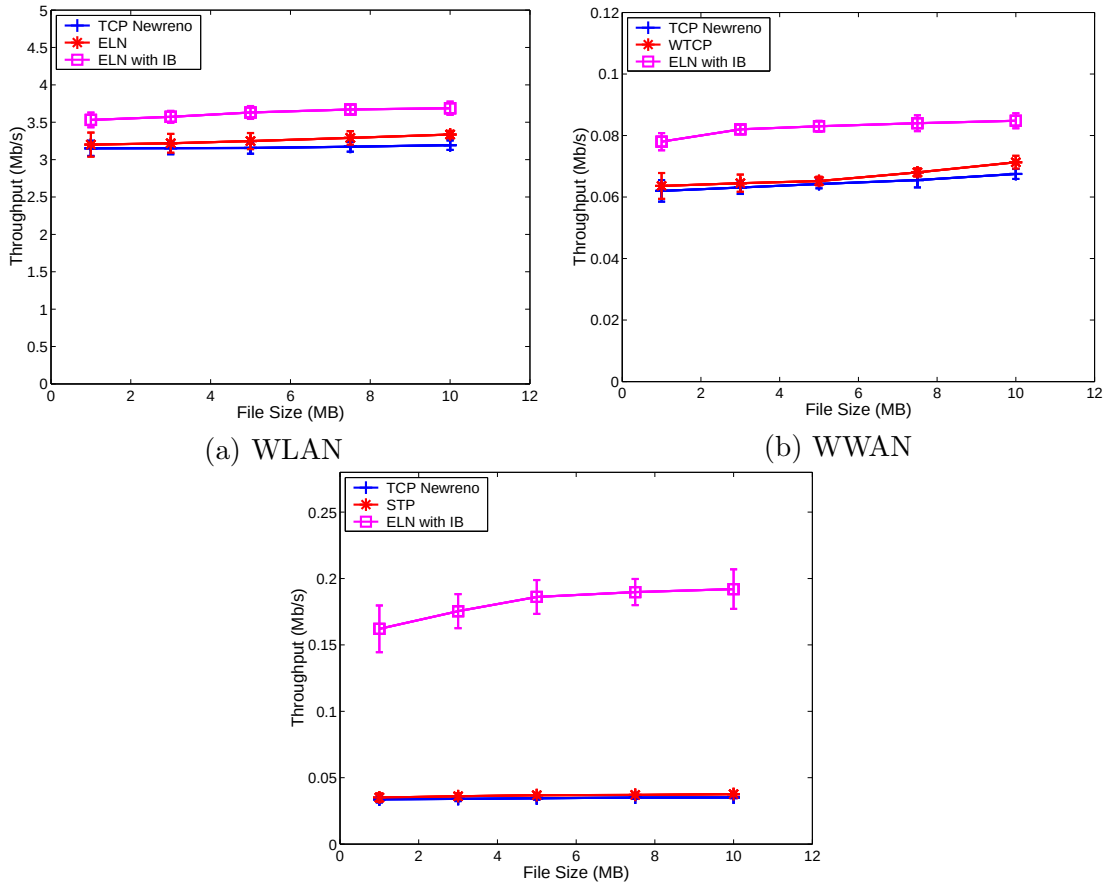


Figure 15: Emulation Results of Infinite Buffering (CIFS)

IB. The IB component on the client side provides virtual buffers, *i.e.* local storage, in user space. It stores data on behalf of the data sender (*i.e.*, SCP server). On the other hand, it supplies stored data to the data receiver (*i.e.*, SCP client) whenever it receives TCP ACKs from it.

In the experiments, a 303 KBytes file is sent from the SCP server to SCP client over the network of 100 Mbps bandwidth and varying RTTs. The tests are performed to learn the impact of RTT on the performance improvement. The results are shown in Figure 17(a). We see considerable improvements are achieved by IB. An interesting observation is IB delivers more improvements with small RTT values. As RTT increases, the actual throughput of SCP also decreases even with IB enabled.

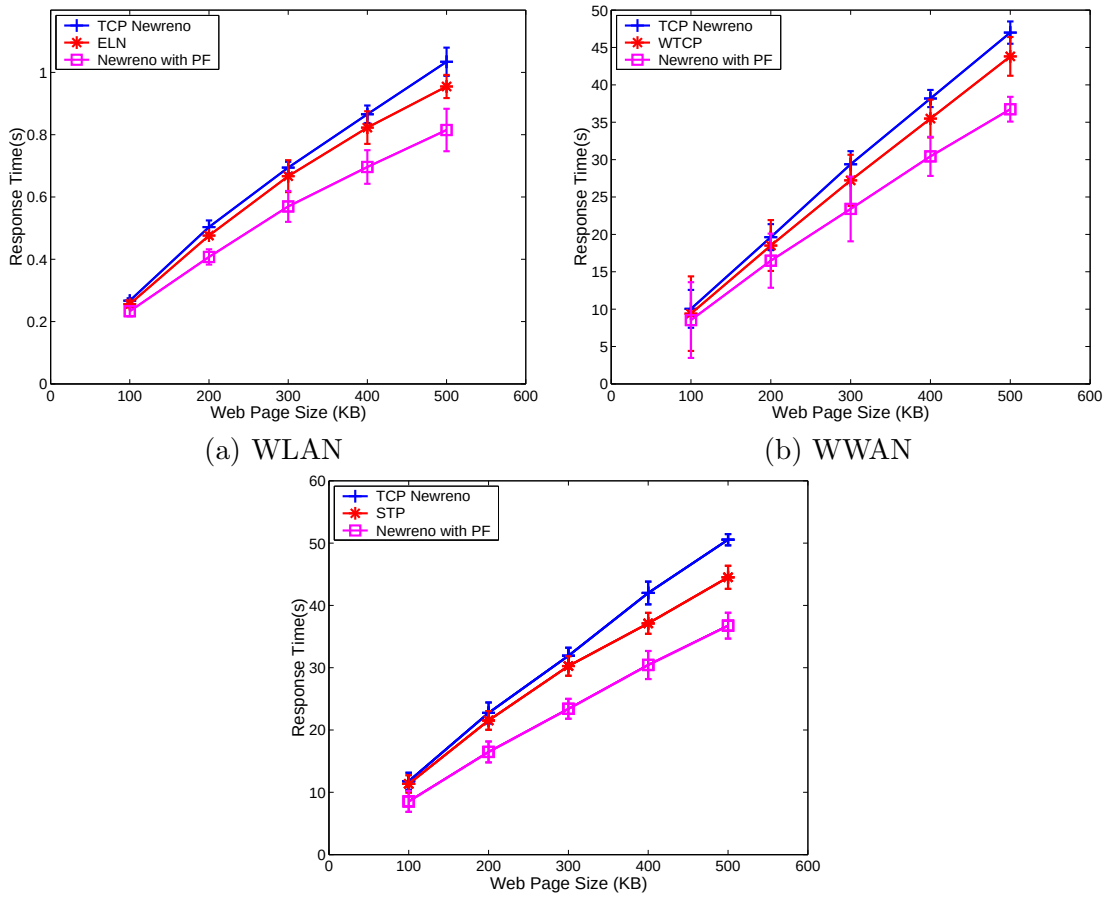
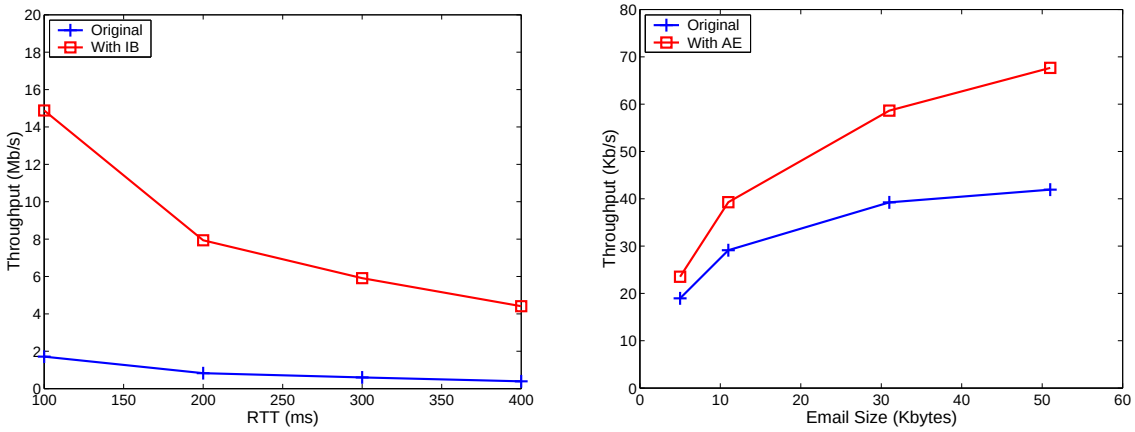


Figure 16: Emulation Results of Prioritized Fetching (HTTP)

2.6.5 Prioritized Fetching

The performance of PF is evaluated with HTTP traffic and results are shown in Figure 16. We consider the Top 50 Web Sites as representatives of typical web pages, and measure their web characteristics. We then use the obtained web statistics to generate the workload. The x-axis in the graphs is the requested web-page size in KBytes, and the y-axis is the response time in seconds for the initial screen. In the figure, it can be seen that as a user accesses larger web pages, the response time difference between default content fetching and PF increases. PF consistently delivers a 15 % to 30 % improvement in the response time performance. PF reduces aggressive traffic volumes by de-prioritizing the out-of sequence fetching of the off-screen objects. Note that PF, while improving the response time, does not improve raw throughput performance. In other words, only the effective throughput,



(a) IB (SCP) (b) AE (SendMail)

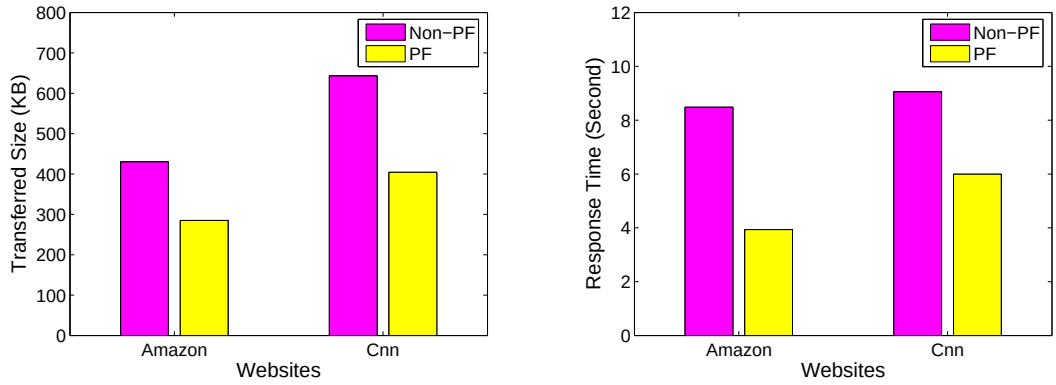
Figure 17: Prototype Results of IB and AE

as experienced by the end-user, increases when using PF.

• *Proof-of-concept Results:* PF is a client-side solution, which does not require any modification at the server side, but requires the integration with the application at the client side. In the prototype, we use WinAPI with Internet Explorer 6.0 on the Windows operating system (running on PC-5).

The PF prototype consists of three main components. The first component is the location-based object prioritization. The current prototype initially turns off the display option of multimedia objects by changing the associated registry values. After the initial rendering is completed without downloading multimedia objects, it calculates the location of all the objects. The second component is the priority-based object fetching and displaying. The current prototype uses the basic on-off model, which fetches the high-priority objects first and then fetches the other objects. If the pixel-size information of the object is inconsistent with the definition in the main document file, the prototype performs the reflow process that renders the entire document layout again. The third component is the re-prioritization. When a user moves the current focus in the application window, PF detects the movement of the window and performs the re-prioritization for the objects that are supposed to appear in the newly accessed area.

The web clients are connected to the Internet and access two web sites: www.amazon.com and www.cnn.com. To highlight the features of PF, we show the results of both transferred size and response time for the first screen in Figures 18(a) and (b), respectively. Note that



(a) Transferred Size (b) Response Time

Figure 18: Prototype Results of Prioritized Fetching (Internet Explorer)

PF can reduce the response time by prioritizing data on the web pages and transferring only high-priority data first. PF sees about 30% improvements on both of these two metrics.

2.6.6 Application-aware Encoding

AE is designed primarily to accelerate email delivery using SMTP and hence we evaluate the effectiveness of AE for SMTP traffic. In the evaluation, emails of sizes ranging from 1 KBytes to 10 KBytes (around 120 to 1200 words) are used. We show the results in Figure 19 where the x-axis is the email size in KBytes and y-axis is the application throughput in Mbps. Varying degrees of throughput improvements are achieved, and in WWAN, an increase of 80 % is observed when transferring a 10 KBytes email. We can see that AE achieves the highest improvement in WWAN due to its relatively low bandwidth.

We also show the effectiveness of AE in terms of compression ratio in Figure 21. In the figure, the results of ten persons' emails using three compression estimators (WinRAR, WinZip and AE) are shown. We can see that WinRAR and WinZip can compress an email by a factor of 2 to 3, while AE can achieve a compression ratio of about 5.

• *Proof-of-concept Results:* The prototype of AE maintains a coding table on either side, and these two tables are synchronized in order to provide encoding and decoding functions. AE monitors the DATA message in SMTP protocol to locate the email contents. The email content is textual in nature, and is expressed using the US-ASCII standard. AE uses the Extended ASCII Codes to provide encoding. We employ a simplified Huffman-style coding mechanism for the sake of operation complexity. The total coding space size is 5,008.

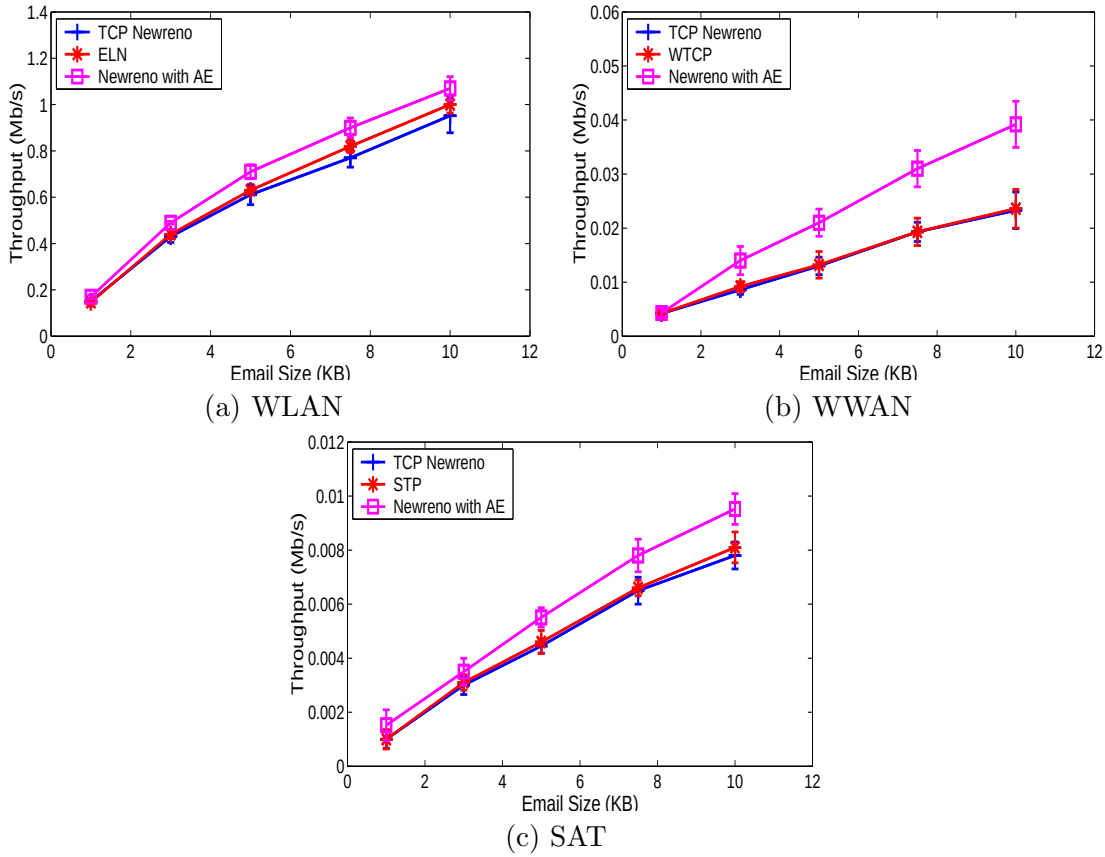
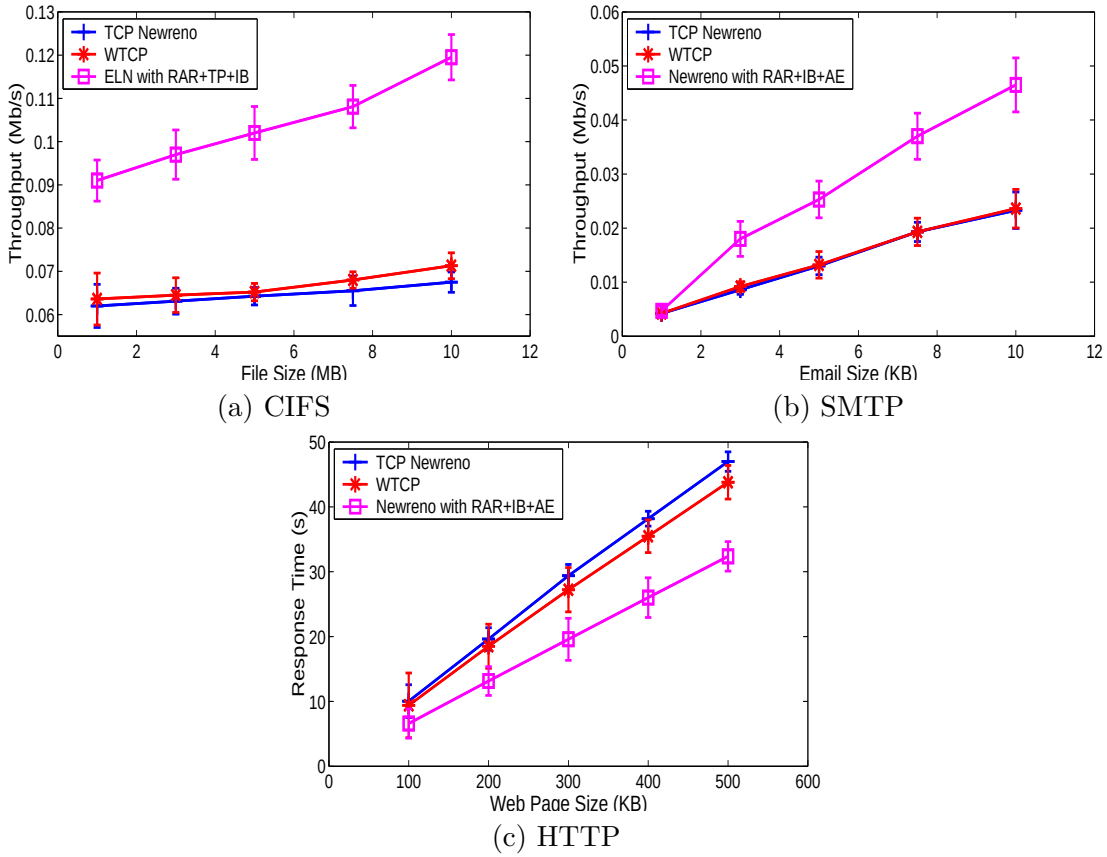


Figure 19: Simulation Results of Application-aware Encoding (SMTP)

The AE component scans an incoming email, and if a word is contained in the coding table, it is replaced by the corresponding tag. If several consecutive words are covered by coding tables, their codes will be concatenated, and necessary padding will be added to the codes to form full bytes. For the words that are not covered by the coding tables, they will stay unchanged with their ASCII representations. Since the email-vocabulary of a user may change with time, AE incorporates a table updating mechanism. AE periodically performs updating operations for every 500 emails. To maintain table consistency between the client and the server, a table synchronization mechanism is employed. Since a user's email vocabulary is expected to change slowly, the AE performs incremental synchronization rather than copying the entire table.

SendMail is used to build the prototype. Purely text-based emails of various sizes are sent from a email client to SendMail server, and the network is configured with 100 ms *RTT* and 50 Kbps bandwidth. The results are shown in Figure 17(b). Every data point is the



(a) CIFS (b) SMTP

(c) HTTP

Figure 20: Integrated A^3 Results in WWAN

average value of five emails of similar sizes.The throughput is improved by 80% with AE.

2.6.7 Integrated Performance Evaluation

We now present the results of the combined effectiveness of all applicable principles for the three application protocols, CIFS, SMTP and HTTP. We employ RAR, TP, and IB on the CIFS traffic in the emulation set-up. For SMTP, the RAR, AE and IB principles are used. For HTTP, the A^3 principles applied are RAR, PF and IB. As expected, the throughput of the applications (CIFS and SMTP) when using the integrated A^3 principles is higher than when any individual principle is employed in isolation, while the response time of HTTP is lower than any individual principle. The results are shown in Figure 20, with A^3 delivering performance improvements of approximately 70 %, 110 %, and 30 % for CIFS, SMTP, and HTTP, respectively.

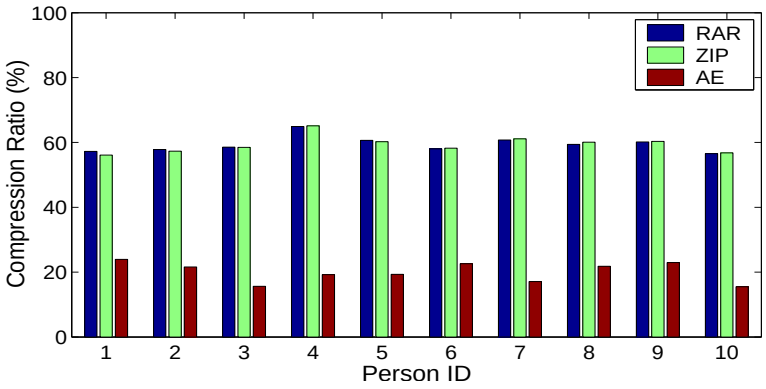


Figure 21: Effectiveness of AE

2.7 Conclusions and Discussion

In this chapter, we motivate the need for application acceleration for wireless-data networks, and present the A^3 solution that is application-aware, but application transparent. We further discuss a few issues in the rest of the section.

• *Insights into A^3 Principles* In this work we present a set of five A^3 principles. We realize that this set is not an exclusive set of all A^3 principles. Hence, we further explore the design space of a general application acceleration framework. Specifically, we argue that the general A^3 framework consists at least five orthogonal dimensions of principles, namely, Provisioning, Protocol Optimization, Prediction, Compression and QoS. In this context, RAR and IB belong to the dimension of Protocol Optimization, TP belongs to Prediction dimension, AE belongs to Compression dimension, and PF belongs to QoS dimension. More principles, as well as more dimensions, are left as part of our future work.

The principles of RAR, IB and AE are application independent, meaning that they can be used to accelerate any application; while PF and TP are application specific and can only help certain applications. We believe such classifications can help gain more insights into the A^3 design, so that the A^3 principles can be incorporated into the design of new applications.

• *TP vs. Opportunistic Pre-fetching* TP is designed to do deterministic pre-fetching rather than opportunistic pre-fetching. Opportunistic pre-fetching techniques aggressively request data that might be used by the end user in the future. For example, some web-access products (e.g. web browsers) pre-fetch data by requesting web contents based on some hints. Our design goal of TP is to do deterministic pre-fetching since otherwise the design will incur overhead incurred by requesting unnecessary contents.

Ensuring deterministic pre-fetching is non-trivial. We now present several approaches to this problem and will explore other approaches in future work. One simple approach is to apply TP only to file transfer operations where users always request a file in its entirety. The second approach is to let TP be fully aware of the application software being accelerated and only pre-fetch data that are definitely needed. In other words, TP can be designed to be sufficiently intelligent so that it can recognize the specific application implementations and

avoid the unnecessary data fetching. For example, CIFS protocol may have various software implementations. Some software may support the range-locking functions, but others may not. If TP is aware of these differences, it can act correspondingly to ensure its deterministic behaviors. But surely, the downside of this approach is the associated design overhead required for such intelligence. In practical deployment, the overhead can be affordable only if the benefits gained are larger than the cost of the overhead. An alternative approach is to relax the strictness of deterministic pre-fetching by tolerating some degree of opportunistic pre-fetching. The corresponding solution is "constrained acceleration". With constrained acceleration, instead of pre-fetching the entire file, TP pre-fetches a "chunk" which is larger than a block. Thus, even if some portion of the pre-fetched chunk is not used, the cost is constrained. The chunk size is defined by acceleration degree, the design of which requires further work. In our proof-of-concept prototype, we adopted such an approach with fixed value of acceleration degree.

• *Preliminary Complexity Analysis* One of the important issues when considering deployment of a technique is the complexity. A^3 can be deployed/realized in multiple ways. For instance, it can be realized in either user space or kernel space, and it can be deployed as either a full platform model or a point model (*i.e.*, $A^3\bullet$). Different deployment or realization models are associated with different degrees of complexity and performance tradeoff.

We now perform certain preliminary complexity analysis in term of lines of codes, memory usage and computation overhead. Our prototype implements A^3 framework in user space and is deployed as a platform solution. (i) The prototype is implemented with about 4.5K lines of *c* codes. Specifically, PF and TP each has about 1K lines of codes, and other elements each has about 600 to 900 lines. (ii) The memory usage varies with different A^3 elements. Specifically, TP uses more memory than other elements since it needs to temporarily hold the returned data corresponding to the accelerated requests, hence the memory size is a function of the acceleration degree and the receiver's consumption rate. IB also stores application data temporally to compensate the receiver's TCP buffer, and the memory size depends on the receiver buffer size and receiver's reading rate. AE needs to allocate a space to store the coding table, so the memory usage is proportional to the

table size. RAR and PF use relatively less memory than other three elements since they maintain little application data and state. (iii) In terms of the computation overhead, we observe little change on the CPU usage when running the prototype. Specifically, AE uses relatively more CPU since it needs to perform data compression. For PF, the CPU usage is higher at the moment of user scrolling up or down since PF needs to re-prioritize the objects.

2.8 Related Work

• *Wireless-aware Middleware and Applications* The Wireless Application Protocol (WAP) is a protocol developed to allow efficient transmission of WWW content to handheld wireless devices. The transport layer protocols in WAP consists of the Wireless Transaction Protocol and Wireless Datagram Protocol, which are designed for use over narrow band bearers in wireless networks and are not compatible with TCP. WAP is highly WWW-centric, and does not aim to optimize any of the application behavioral patterns identified earlier in the chapter.

Browsers such as Pocket Internet Explorer (PIE) [26] are developed with capabilities that can address resource constraints on mobile devices. However, they do not optimize communication performance, which is the focus of A^3. Work in [89] aims to save bandwidth and power by adapting the contents based on user semantics and contexts. The adaptations, however, are exposed to the end-applications and users. This is different from the A^3 approach which is application-transparent.

The Odyssey project [90] focuses on system support for collaboration between the operating system and individual applications by letting them both be aware of the wireless environment, and thus adapt their behaviors. Comparatively, A^3 does not rely on the redesign of the OS or protocol stack for its operation, and is totally transparent both to the underlying OS and the applications. The Coda file system [106] is based on the Andrew File System (AFS), but supports disconnected operations for mobile hosts. When the client is connected to the network, it *hoards* files for later use during disconnected operations. During disconnections, Coda emulates the server, serving files from its local cache. Coda's

techniques are specific to file systems, and require applications to have changed semantics for the data that they use.

• *Related Design Principles* Some related works in literature have been proposed to accelerate applications with various mechanisms [57, 58]. We present a few of them here, and identify the differences vis-a-vis A^3. *(i) TP-related:* In [56], the authors propose to "upload" clients' task to the server side, thus eliminating many RTTs required for applications like SMTP. This approach is different from the A^3 approach in terms of application protocols applied and the overall mechanism. *(ii) RAR-related:* Mechanisms like FEC (Forward Error Control) use error control coding for digital communication systems. A link-layer retransmission approach to improve TCP performance is proposed in [96]. Another work [116] proposes aggressive retransmission mechanism to encourage legitimate clients to behave more aggressively in order to fight attack against servers. Compared to these approaches, A^3 only applies RAR to control messages in application protocols, and it does so by retransmitting the control messages when a maintained timer expires. We present arguments earlier in the chapter as to why protecting control message exchanges is a major factor affecting application performance. *(iii) PF-related:* To improve the web-access performance, tremendous work have been done [103], [26], [19], [42]. Work in [89] proposes out-of-order transmission of HTTP objects above UDP, and break the in-order delivery of an object. However, unlike the A^3 framework, it requires the cooperation of both client and server sides. *(iv) IB-related:* The mechanisms such as [101], TCP Performance Enhancing Proxy (TCP PEP) [28] are proposed to shield the undesired characteristics of various networks, particularly wireless networks. IB is different from these approaches, which aim at fully utilizing the network resources by removing the buffer length constraint. IB specifically applies to applications with bulk data transfer, such as FTP, and is meant to counter the impact of flow control. Some work also observe that applications suffer from poor performance over high latency links due to flow control. For example, [101] proposes to change the SSH implementations to remove the bottleneck caused by receive buffer. *(v) AE-related:* Companies like Converged [6] provide application-aware compression solutions through compressing the data for some applications based on priority and application

nature. These mechanisms share the property of being *application aware*, meaning only a subset of applications will be compressed. However, AE has the property of being *user-aware*, that is, taking into consideration user-specific information, and thus can achieve better performances.

- *Commercial WAN Optimizers* Several companies, such as Riverbed [29] and Juniper [14], sell WAN-optimization application-acceleration products. However, (i) Almost all the commercial solutions are proprietary ones; (ii) The A^3 principles such as RAR, IB, AE and PF are not seen in commercial solutions; and (iii) Many of the techniques used in commercial solutions, such as bit-level caching and compression, are hardware-based approaches, and require large amounts of storage. These properties render the commercial solutions inapplicable for environments where easy deployment is required. A^3 is a middleware approach requiring small amounts of storage.

CHAPTER III

ACCELERATING PEER-TO-PEER APPLICATIONS WITH MOBILE HOSTS PARTICIPATING IN NETWORKS: CHALLENGES AND SOLUTIONS

3.1 Summary

Peer-to-peer (P2P) data networks dominate Internet traffic, accounting for over 60% of the overall traffic in a recent study. In this work, we study the problems that arise when mobile hosts participate in P2P networks. We primarily focus on the performance issues as experienced by the mobile host, but also study the impact on other fixed peers. Using BitTorrent as a key example, we identify several unique problems that arise due to the design aspects of P2P networks being incompatible with typical characteristics of wireless and mobile environments. Using the insights gained through our study, we present a wireless P2P (wP2P) client application that is backward compatible with existing fixed-peer client applications, but when used on mobile hosts can provide significant performance improvements.

3.2 Introduction

Over the last few years, peer-to-peer (P2P) data sharing applications have experienced an explosive growth. In recent years, a staggering 60% of the Internet data traffic constituted of P2P file sharing [25]. While copyright concerns had earlier brought down popular P2P applications such as Napster, several content owners and providers have of late started embracing new types of P2P technology that have come to stay [3]. P2P data sharing is now not only being considered as a means for consumer level data exchange, but also as a viable means of delivering data from professional content producers to their consumers ([2, 11, 17]). Hence, the dominance of P2P traffic in the Internet is expected to continue to grow in the near future.

With P2P data sharing applications securing a dominant position in the Internet landscape, a natural question to ask is: *what is the performance of mobile users when participating in P2P data sharing?* The question is significant because of the following two reasons. First, as with any Internet application with emerging or established popularity, wireless and mobile users are increasingly adopting P2P data sharing applications on devices such as laptops and PDAs [67]. With the number of users using wireless technologies for Internet access growing rapidly, it is inevitable that P2P data sharing over wireless and mobile environments will assume significance. Second, there are several efforts underway to deliver P2P data sharing as the next killer application for mobile devices ([18,22]). Initial instantiations of such efforts focus on sharing of ring-tones and music files, but are expected to evolve into other types of content including video. Thus, in both contexts, it is important to consider what levels of performance such users will enjoy, and investigate the match-ups between the typical designs of P2P data sharing applications and the characteristics of wireless and mobile environments.

Thus, in this work, we first investigate the following question: *what is the performance of a mobile user in a wireless environment using a P2P data sharing application?* As a corollary, we also investigate the following question: what is the performance of a fixed peer in a P2P network when using a mobile host as a corresponding peer? In answering the above questions, we find that several of the fundamental design principles and peculiarities used in P2P data sharing applications are inconsistent with the key limiting characteristics of typical wireless and mobile environments. Briefly, some of these issues include: (i) P2P applications, unlike in typical scenarios where a mobile host functions as a client, create a scenario requiring the mobile host to function as a server. This raises several implications including mobility and Power Save Mode (PSM) on server sides. (ii) P2P data sharing uniquely involve *simultaneous* bi-directional data transfer. This consequently results in the use of bi-directional TCP, a form of TCP not studied extensively for wireless environments. (iii) P2P data networks, by virtue of being almost entirely supported by end-hosts, typically use incentive-based performance delivery. Because of this, a mobile host that has uploaded more data is provided with higher performance. Such a mechanism exposes issues when

applied as-is to a wireless and mobile environment. (iv) While incentives encourage P2P users to stare data longer, P2P data fetching is also adapted to increase the uniquely shareable data available at a user. One such approach is performing random or rarest-first fetching. However, such techniques have severe implications to the mobile user, especially during disconnections. With a real-life P2P data sharing network, we study the performance of mobile users with respect to the above issues.

Using insights gained through the aforementioned study, we present a deployable solution suite called *w*P2P that addresses the issues by only changing the P2P application at the mobile host. *w*P2P uses techniques transparent to the fixed peer, but uniquely relevant to the specific issues pertaining to wireless and mobile hosts functioning in a P2P data network. Specifically, *w*P2P uses a combination of four design principles including Age based Manipulation, Incentive Aware operations, Mobility-aware Fetching, and Role Reversal. We evaluate the effectiveness of *w*P2P through both simulations and prototyping, and the evaluations show that significant throughput performance improvements can be achieved for mobile hosts and fixed peers, when using *w*P2P at the mobile hosts. Thus, the contributions of this work are twofold:

- We consider the specific scenario of mobile hosts participating in P2P data sharing networks, and investigate performance issues such hosts face due to the unique design elements embedded in typical P2P applications. Using real-life P2P experiments, we identify these design elements that when combined with the unique characteristics of wireless environments render the performance delivered to mobile users sub-par.

- We present a deployable solution called *w*P2P that is required to be instantiated only at the mobile host and can deliver enhanced P2P data sharing performance to mobile users. Using a prototype implementation, we characterize the performance of *w*P2P and show that considerable improvements can indeed be achieved.

The rest of the chapter is organized as follows: Section 3.3 presents the scope of this work and describes key background material. Section 3.4 presents in detail the motivation results that show the limiting performance that existing P2P application design imposes on mobile users. Section 3.5 presents the key design principles of *w*P2P and the realizations of the

principles. Sections 3.6 presents the evaluation results with both simulation and prototype implementation. Finally, Section 3.7 discusses related work and Section 3.8 concludes the work.

3.3 Background and Scope

In this section, we describe the scope of this work, as well as the necessary backgrounds of P2P networks.

3.3.1 Scope of this work

• *P2P Networks:* While there are several forms of P2P networks ranging from those that help in computing (*e.g.*, grids) to those that help in communication (*e.g.*, Skype) to those that help in data-sharing, this work is entirely focused on P2P data sharing networks. Data sharing P2P networks are primarily used for sharing files containing audio (*e.g.*, mp3 files), video (*e.g.*, mpeg2 files), or data (*e.g.*, Linux distributions). Examples of such networks include BitTorrent [3], eDonkey [7], Gnutella [13], and FastTrack [10]. Interestingly, the above four example networks account for almost all the P2P traffic in the Internet and together constituted more than 60% of the Internet traffic in recent studies [25]. Specifically, traffic carried using BitTorrent account for over 30% of the overall Internet traffic. Also, video files account for more than 62% of P2P traffic.

In this work, while we identify characteristics of P2P networks that are generically applicable to all four of the above networks, we use BitTorrent as the primary example for all discussions, experiments, and trials. However, as necessary we also step back and investigate relevance of our discussions and interpretations for the other networks as well. We believe that this choice of BitTorrent as the key representative is justifiable from multiple standpoints including its dominance in terms of traffic carried, and its relative sophistication. Example (and popular) clients that implement BitTorrent are Linux-based Azureus and Windows-based BitSpirit.

• *Data Types:* While BitTorrent is primarily used for sharing video files that are large in size (*e.g.*, few hundreds of megabytes or more), a non-trivial amount of its traffic also includes audio (*e.g.*, few megabytes or more) and software distribution. Hence, in this work,

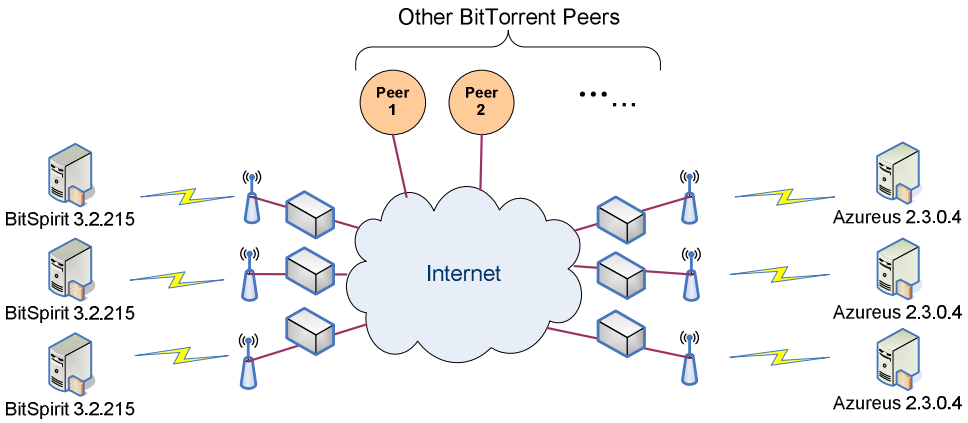


Figure 22: Network Testbed for P2P Evaluation (All six BitTorrent peers are inside Georgia Tech campus)

we consider file data sizes that represent both ends of the above spectrum.

- *Wireless Technologies and Mobile Devices:* Measurement studies conducted recently (such as [67]) observe far more wireless and mobile users on the network than ever before. While mobile users with any type of wireless access can participate in P2P networks, the access technology typically used is wireless LANs (WLANs). This is because of both the higher bandwidths available on WLANs, and the relatively lower or no-cost models associated with such networks. Hence, we consider WLANs for the wireless environment in this chapter. However, we revisit this assumption later in the chapter to discuss implications of our inferences and proposed strategies for other types of wireless environments. Similarly, while other mobile devices such as PDAs and IP-enabled cell phones become more practical for assuming membership in P2P networks, we primarily consider laptops as the mobile device in this work.

- *Metrics:* We consider throughput performance as the main metric in our evaluation and optimization considerations. While the focus is more on the question: what is the performance of a mobile host when it participates in a P2P network? We also consider a corollary question: what is the performance of a fixed peer when it uses a mobile host as a peer to download data from?

3.3.2 BitTorrent

BitTorrent, like other P2P data sharing protocols, uses peers that have downloaded a certain content as the sources for the content subsequently for other peers that need the same content. Any peer that implements the BitTorrent protocol can participate in a BitTorrent network. We now outline some of the key elements of the BitTorrent protocol that are relevant to the focus of this work.

- *Torrent file:* Any peer that wants to share a file through the BitTorrent network creates a ".torrent" file that consists of some meta data information (*e.g.*, certificates for different portions of the file that downloading clients can use to check for the validity of downloads) and the address of the *tracker* that will act as the directory server for the file.

- *Tracker:* The tracker is the entity that maintains, for any given file that it tracks, all current peers that have the file either in its entirety or in parts. When it receives a request from a client for a specific file, it furnishes the client with the addresses of the peers associated with the file. The list of peer addresses is updated periodically to accommodate peers leaving and joining the network. Since peers download parts of the file out of sequence, two peers that start downloading at approximately the same time can still provide data to each other quickly.

- *Swarm:* All peers currently connected to each other in the process of downloading a particular file form a swarm for that file. From the standpoint of a single client, it uploads to some members in the swarm and downloads from some members in the swarm. Because of the incentive policy discussed later in this section, a downloading client is likely to be uploading to members it is downloading from.

- *Tit-for-Tat:* BitTorrent uses a tit-for-tat incentive policy that controls the upload rate of one peer to other peers based on the download rate the peer enjoys from that other peer.

- *Rarest-first fetch:* BitTorrent peers do not fetch parts of the file in sequence. Instead, each peer picks the *rarest* of the blocks (in terms of the number of peers in the swarm that have the block) preferably to download. This ensures that the rarest blocks of a file are propagated in the swarm faster, reducing bottlenecks at the few peers that have the block and increasing the availability of those blocks if the peers that have them shutdown.

• *Seeds and leeches:* Seeds are peers that have a complete copy of a file, and leeches are peers that have a partial copy of the file, which typically means that the peers are downloading other parts as they are uploading the parts they already have.

Thus, when a peer wants to download a particular file, it downloads the torrent file, contacts the tracker specified, receives a list of addresses to contact, and joins the swarm for downloads. As soon as it receives blocks of the file, it also begins uploading to other peers that require those blocks. When the entire file is downloaded, the peer may decide to leave the swarm or stay back as a seed.

3.3.3 Other P2P Data Networks

While BitTorrent has a centralized aspect to its operation in the form of the tracker, Gnutella and FastTrack belong to the category of decentralized P2P networks where peers search for content in directories distributed around the network. Gnutella and FastTrack albeit differ in the way peers act as content directories. Any peer can act as a host for the directory of content in Gnutella Network. On the other hand FastTrack distinguishes between peers that have slow connectivity to those peers (*i.e.*, super nodes) that have faster and stable connections. FastTrack achieves robust content search service by using the ultra-peer architecture where only super nodes act as content directories.

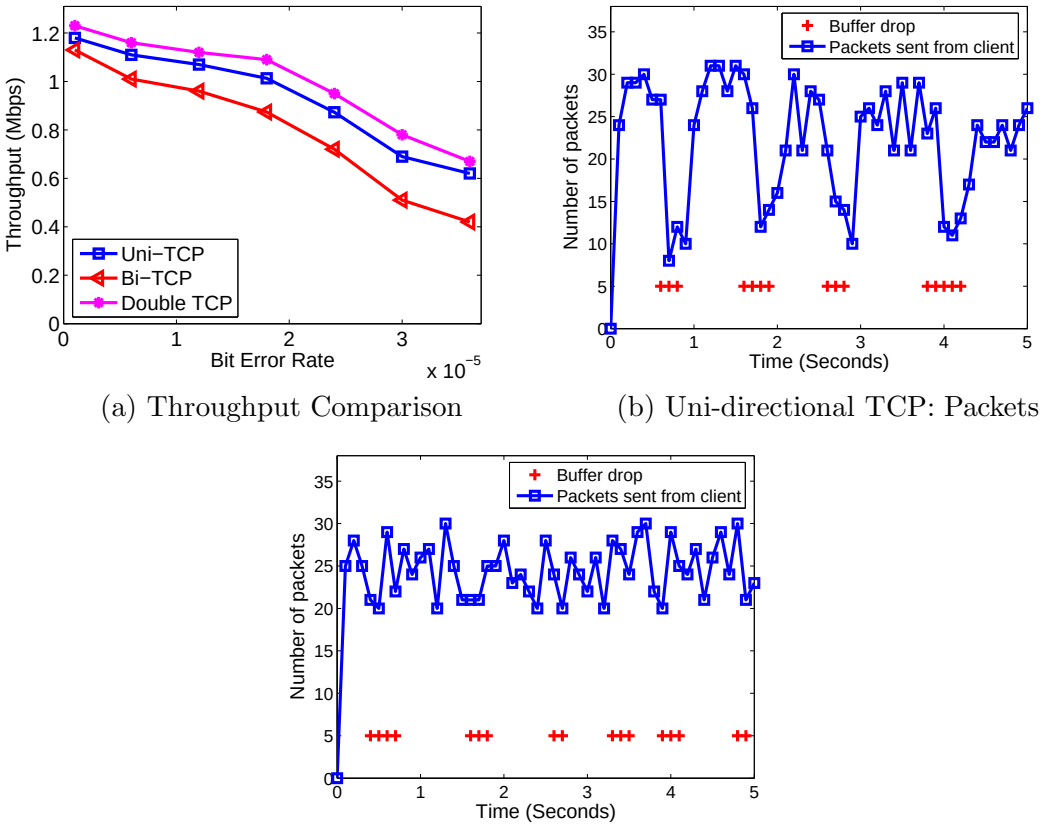
Although the above mentioned P2P networks vary in the way content search is performed, all of them rely on direct downloading and uploading of content. Single HTTP connections are employed for data transfer from the content source to the requesting peer. The third generation of P2P data networks use multiple downloads from several peers to accelerate the content fetching process, and networks like eDonkey (and BitTorrent) fall under this category. An important difference between eDonkey and BitTorrent is the way multiple fetches are performed for a particular content. eDonkey performs contiguous downloads from multiple peers as opposed to random fetches performed by BitTorrent.

3.4 Motivation

In this section, we use experiments performed on a BitTorrent network to study the performance of a mobile host in a P2P data network. We identify several design characteristics

of P2P data networks that are incompatible with typical characteristics of a wireless environment. We also use the insights gained as the basis for the design of the *w*P2P solution presented later in the chapter. While we present all the discussions in the context of BitTorrent, we revisit the implications of the discussions on the other P2P networks at the end of the section.

3.4.1 Testbed & Methodology



(a) Throughput Comparison

(b) Uni-directional TCP: Packets

(c) Bi-directional TCP: Packets

Figure 23: Impact of Bi-directional TCP

The main network testbed used in the experiments is shown in Figure 22. The testbed under our control consists of six BitTorrent peers, three of which run the Azureus client 2.3.0.4 on Linux, and other three peers run the BitSpirit 3.2.215 client on Windows. All six peers are inside Georgia Tech campus. They are all connected to Internet through wireless LAN and thus participating in larger P2P networks. To evaluate the performance under various network conditions, we also introduce an emulator for each BitTorrent client. The

emulators run Linux-based Ubuntu OS with kernel modules to mimic the wireless-related characteristics including bit error rate, mobility, delay and bandwidth.

We identify totally six issues when mobile hosts participate in P2P networks, and each of them will be elaborated in this section. Based on where the particular issue arises, we broadly classify these issues into three categories: Issues with Bi-directional traffic, Downloader-side issues, and Uploader-side issues.

3.4.2 Issues with Bi-directional traffic

Issues with Bi-direction traffic are caused by the data exchange between two peers over wireless media. Two issues fall into this category: Bi-directional TCP and Uploads based Incentives.

3.4.2.1 Bi-directional TCP

As described in Section 3.3, most peers in BitTorrent upload and download at the same time, and because of the tit-for-tat policy used by BitTorrent, several of the uploads are to peers from which downloads are being done. Hence, *it is common for data to be exchanged simultaneously between peers in both directions.* Given that BitTorrent (or for that matter the other P2P applications) uses TCP, and that TCP is inherently designed to be a bi-directional protocol, BitTorrent uses TCP in its true bi-directional mode. Thus, a single TCP connection is used to transfer data in both directions between the peers. While TCP is designed to be a bi-directional transport protocol, few extensive studies of its behavior have been performed. More importantly, in the context of this work, very little is understood about the behavior of bi-directional TCP in a wireless environment.

In this context, we identify two issues with the use of bi-directional TCP by BitTorrent. These two issues are centered around the behavior of TCP with respect to *ACK piggybacking* and *fast-retransmit* under bi-directional conditions.

• *ACK Piggybacking:* When bi-directional TCP is used, TCP ACKs in the reverse path are piggybacked on the data packets being sent in the reverse direction to avoid sending separate ACK and data packets. In a wireless environment, where random errors rates can be non-trivial, this potentially has an adverse impact on the connection performance.

More specifically, when ACKs are piggybacked on data packets, the effective "length" of the ACKs is longer than if they were sent as *pure* ACKs (non-piggybacked). Hence, for the same bit error rate (BER) in a wireless environment, the effective packet error rate (PER) for the ACK traffic is larger. This in turn results in more ACK packets being lost on the reverse path just because they were piggybacked. Specifically, for a given packet of size S bytes, assuming the uniform BER is b, the PER can be calculated as $1-(1-b)^S$.

While it is true that TCP uses cumulative ACKs, and hence is relatively robust to ACK losses, there still is a negative impact in terms of the overall throughput enjoyed by a connection in the presence of higher number of ACK losses. More importantly, in a P2P network peers typically have a large number of TCP connections ongoing even for a single swarm (BitTorrent trackers typically provide addresses of 50 peers in response to a request, but the overall swarm size can easily grow to numbers greater than 1000, as we observed from our experiments), resulting in the average congestion window size of a TCP connection to be relatively small. And, it is for connections with small congestion window sizes that a higher ACK loss rate can result in a non-trivial degradation in throughput. Thus, the download rate for a TCP connection from a particular peer will be smaller just because of ACKs being piggybacked in the reverse direction.

We setup experiments to obtain the results with respect to the use of uni-directional TCP and bi-directional TCP. The network setup is as follows. To study the impact of bit error rate, we implement a kernel module with NetFilter [21] to probabilistically drops packets based on packet lengths and assumed bit error rates. To simulate the special feature of shared-media in 802.11, we implement a shared-bandwidth emulation module which embeds at the NF_IP_FORWARD point inside the NetFilter framework. The maximum bandwidth is set to 300 KB per second (i.e., 2.4 Mbps). [1]

We consider three scenarios: single uni-directional TCP, single bi-directional TCP and double uni-directional TCP. We manipulate the shared data file on the testbed to create all scenarios. (i) For uni-directional TCP, we ask a peer to seed a file, and another peer

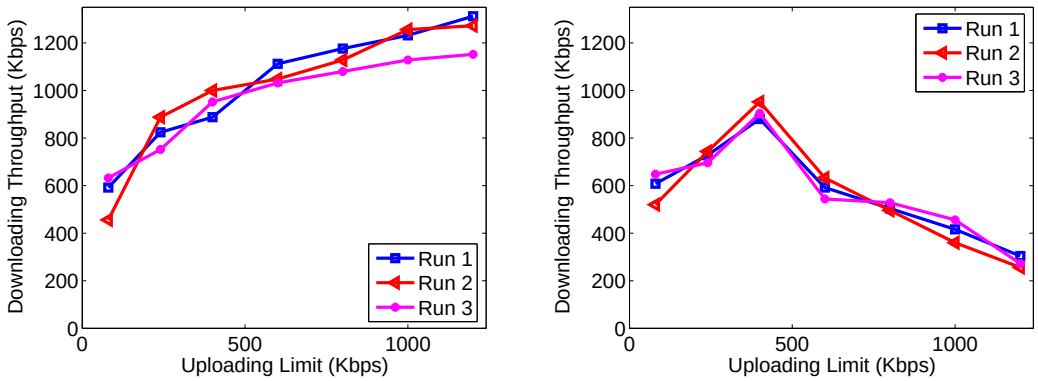
[1]Note that though we could perform experiments by relying on the "ture" shared-media of the WiFi networks, experimenting under more-controlled environments help filter out other irrelevant factors such as interferences from other WiFi networks.

purely downloads the file. (ii) For bi-directional TCP, we use two other peers to download the file for a period of time. After these two peers have downloaded about half of the file, we remove other peers and only keep the two peers. Since these two peers each has a random portion of the file, they begin to exchange data between them. We observe that the two peers are sending and receiving data on a single TCP connection. For this scenario, we only consider the throughput of one direction (i.e., the larger one). (iii) For double uni-directional TCP, we first set up the scenario of single uni-directional TCP, then creates another uni-directional TCP along the opposite direction with Iperf. Given two peers A and B, A is downloading from B with BitTorrent, and concurrently A is also uploading to B with Iperf. For this scenario, we consider the aggregate throughput.

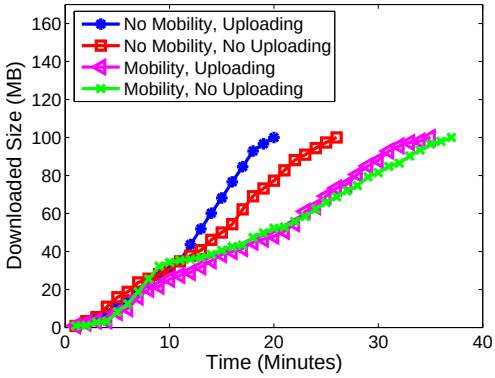
Figure 23(a) presents results of the download rate experienced by a peer under varying conditions of bit error rate. Each data point is the average of five 10-minute runs. We observe that the aggregate throughput of double uni-directional TCPs result in highest throughput, which can be easily explained by the higher usage of the bandwidth with two connections.. The bi-directional TCP has the lower throughput than uni-directional TCP, which indicates the impact of ACK piggybacking. What is more interesting is that the degradation in throughput performance with increasing BER is faster in the case of bi-directional TCP. In addition, while it might appear that peers are better off not uploading for this reason, recall that the tit-for-tat policy used by BitTorrent will render such a strategy ineffective.

• *Fast Retransmit:* The second issue with BitTorrent using bi-directional TCP occurs during congestion. TCP's fast retransmit mechanism is based on the arrival of DUPACKs (duplicate ACKs) at the sender. When bi-directional TCP is used, TCP specifications stipulate that DUPACKs should not be piggybacked. The reason for this stipulation is that otherwise the sender has no way of knowing whether piggybacked DUPACKs are due to losses, or due to the receiver sending multiple data packets between two data packet arrivals (and hence piggybacking the same ACK sequence number on those data packets).

Hence, a TCP receiver will send *only* pure ACKs when DUPACKs have to be transmitted due to losses. While this design has no issues in a wired environment, it results in problems



(a) Wired Network (b) Wireless Network



(c) Downloading Process of a 100 MB File

Figure 24: Effect of upload traffic on downloads (a,b), Effect of Incentive and Mobility (c)

when performed as-is in a WLAN environment. This is because, when congestion has occurred in the WLAN resulting in a packet drop, the DUPACKs sent back will be explicitly de-coupled from the data stream in the reverse path, thus *actually increasing the number of packets in transit on the wireless leg.* While it is true that the TCP sender, when it receives the DUPACKs will reduce its number of outstanding packets in the network by half according to TCP specification, the new transmissions of pure ACKs essentially offsets the decrease in the number data-packets. Hence, as far as the wireless leg is concerned, the number of packets in transit stays approximately the same even after a congestion event.

Figures 23 (b,c) show the number of packets sent by a client on the wireless leg with time for two BitTorrent connections, one uni-directional TCP and one bi-directional TCP, respectively. For the connection using uni-directional TCP, the client is sending data to another BitTorrent peer. For the connection using bi-directional TCP, the client is both

sending to and receiving from another peer. The congestion events are indicated as buffer drops in the figure as well. It can be observed that while the number of packets on the wireless leg in general decreases after congestion events for the uni-directional connection, this is not true for the bi-directional TCP connection. This "mis-behavior" can potentially result in performance degradation for the connections due to deviation from the intended behavior of TCP's congestion control.

We have thus far discussed the negative impact of using bi-directional TCP connections. While it is true that the above problems will be issues for any application that uses simultaneous data transfer with bi-directional TCP, it is noteworthy that P2P data networks are perhaps one of the few instances (if not the only) where such bi-directional data transfer does happen simultaneously and in large volumes. Applications (or protocols) such as http or secure-shell, while using bi-directional TCP, do not exercise the bi-directionality the same way as P2P networks do. For example, both the above applications have distinct request-response patterns that render simultaneous data transfers at high volumes unnecessary in both directions.

3.4.2.2 Uploads-based Incentives

The tit-for-tat incentive mechanism in BitTorrent encourages higher rates for uploads to enjoy better download rates. In a wired environment, it can be shown that peers enjoy their best download rates when their upload rates are high. Figure 24(a) shows the aggregate download rates of 4 simultaneous tasks (i.e., top 4 torrents of Fedora 10 [32]) as a function of the upload rate limit in a wired setting. The network is set up with the kernel-module-based bandwidth emulator we discussed before. Specifically, the bandwidth is capped to 2.4 Mbps for both uplink and downlink. We show the results of three separate runs, and each data point shows the result of 1-hour running. We observe that the download rate is an increasing function of the upload limit. Since with wired links, the uplink and downlink physically do not affect each other, we believe that the results are caused by the incentive mechanism embedded in BitTorrent. This inference is further supported by variable software manuals and user-experiences, most of which state that the recommended uploading rate should be

typically around 80% of the upload capacity. The remaining 20% capacity is reserved to allow for the pure TCP ACK traffic on the reverse path for other connections.

For a wireless environment, however, the relationship between the enjoyed download rate and the upload rate limit changes. Using the same kernel module but with a shared bandwidth-cap between uplink and downlink, we show the aggregate download rate of the same four tasks in Figure 24(b). As shown, while the download rates initially increase with higher upload rates, beyond a much smaller upload rate (400Kbps, much less than 80% in wired networks) the download rates actually drop. This is due to the *shared* channel nature of the wireless link, where the uploads and downloads are contending for the same wireless channel bandwidth. This is in contrast to a wired setting where the uploads and downloads do not share the same bandwidth resources. The same figure also demonstrates that shutting down the upload is not a solution either as the tit-for-tat strategy of BitTorrent will kick-in punishing the peer with low download rates.

This problem of uploads contending with the downloads also goes beyond the incentives mechanism in BitTorrent. In a wired environment, peers have a relatively low dis-incentive to continue as a seed once the downloads are completed with the argument being that upload bandwidth is anyway largely unused under most conditions except for TCP ACK traffic. Thus, peers in wired networks are more willing to upload other peers by acting as servers. However, in a wireless setting, a mobile peer functioning as a seed can potentially impact its *download rates for other non-P2P applications* without any direct counter-benefit for the user. While this contention between P2P uploads and other application downloads is true even when the mobile peer is a leech, it can be argued that the mobile user is at least receiving the benefit of the tit-for-tat scheme. However, when the wireless peer converts to a seed, this advantage no longer remains. Hence, any viable solution to motivate mobile peers to continue as a seed has to ensure that the uploads do not negatively affect the downloads of other applications.

3.4.3 Downloader-side issues

Downloader-side issues occur in the downloading process on P2P clients. These issues include: Incentives and Mobility and Rarest-first Fetches.

3.4.3.1 Incentives and Mobility

The tit-for-tat mechanism in BitTorrent is associated with a unique identifier for the peer called the *peer-id*. The peer-id is typically constructed as either a function of the IP address of the host and a random value, or simply as a function of a mobile host specific random value. The peer-id is regenerated every time fetch tasks are re-initiated. Thus when a mobile host experiences a hand-off and receives a new IP address, the ongoing tasks are terminated and the tasks are re-initiated[2] thus generating a new peer-id. However, since the peers track the *goodness* of corresponding peers based on the peer-id, this results in the mobile peer losing all the credit it has built with its corresponding peers.

Figure 24(c) shows the effect of incentives on the downloaded sizes of a 100MB file as a function of time. Under the no-mobility scenario, we observe that the download size is lower when there is no upload traffic. This is the normal incentive behavior. However when we introduce mobility by switching IP address every one minute, we see that the incentive mechanism is rendered ineffective. Not only is the actual download size lower than the no-mobility case, there is marginal difference between the download rates with or without uploading. This is because every time the IP address changes, tasks are re-initiated and thus the host acts as a new peer without any previous incentives. Thus, the mobility of a peer can have an adverse impact on the incentive mechanism of a P2P network.

3.4.3.2 Rarest-first Fetches

As outlined in Section 3.3, BitTorrent employs a rarest-first fetching paradigm. This results in any snapshot of the downloaded content for a file not having any significant "in-sequence" data from the head of the file till a large percentage of the file download is completed. Many media formats, on the other hand, allow for partial playback of content provided the partial

[2]We assume here that mobile IP [97] is not used to handle mobility.

information is in sequence. For example, for an MPEG file of a 2 hour video, the download of the first 30 minutes worth of the video will still allow for a playback of that part of the video. Figure 25(a) shows the *playable* fraction of two files being downloaded with increasing fraction of the actual downloads using rarest fetch. It can be observed that until a large percentage of the whole file download is complete, a significant percentage of the file still remains unplayable. For 5 MB file, even with a 60% download fraction, less than 20% of the file remains playable. In fact, for the 100 MB file size scenario, more than 90% of the file size needs to be downloaded to playback the first 5% of the video.

While this property of BitTorrent is an irritant even for a fixed peer, it is justifiable for two reasons: (a) this enables the peer to contribute well to the P2P network as it is likely to have blocks that are different from those at other peers; and (b) fixed peers do not have to concern themselves with wireless disconnections thus ensuring that the downloads will eventually complete and will not be in vain.

However, for a mobile peer, this property can have more serious implications. In the example of the 100 MB file, if the mobile peer gets disconnected from its wireless network (and remains so) after 90% of the file has been downloaded, the user still cannot playback more than 5% of the content. Furthermore, the 90% of the file size downloaded thus far using the rarest-first algorithm in the interest of the well being of the rest of the P2P network cannot be served back to the P2P network anyway because of the disconnection.

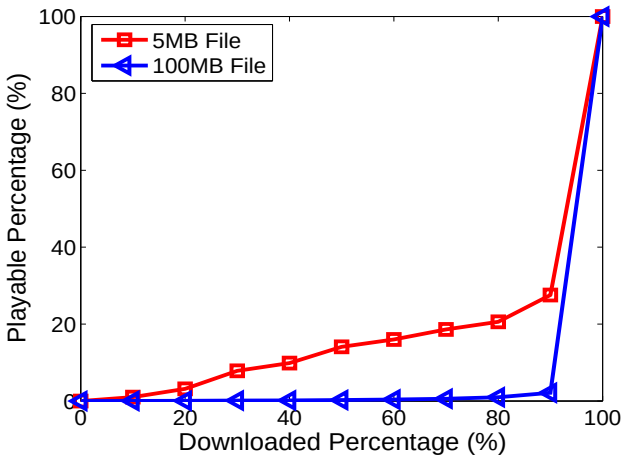
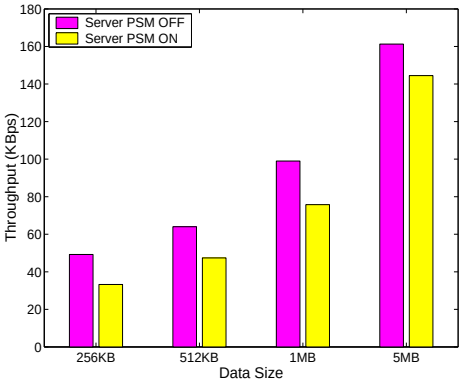
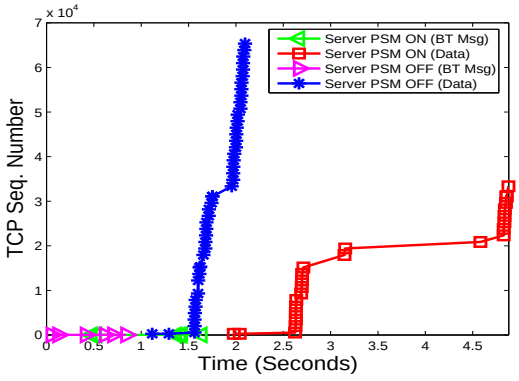


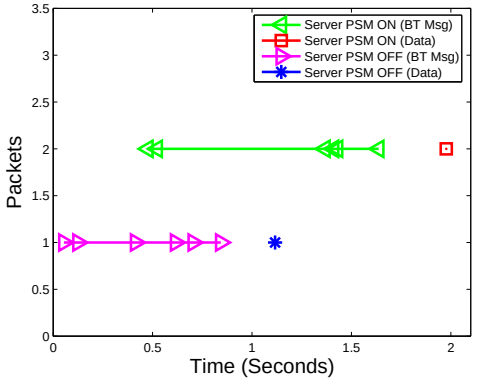
Figure 25: Impact of Rarest-first Fetching



(a) Effect of PSM (Client PSM OFF)



(b) Progress of packet sequences



(c) Microscopic progress

Figure 26: PSM on Server side

3.4.4 Uploader-side Issues

Uploader-side issues are caused by the fact that P2P clients are also serving as "servers" in the sense that they supply data to other clients. We identify two issues in this category: Power-saving Mode and Server Functionality, and Server Mobility.

3.4.4.1 Power-saving Mode and Server Functionality

Most mobile devices employ a power-saving mode (PSM) of operation to shutdown their radios and conserve energy. In the 802.11 PSM operation, the radio goes to sleep when there is no data to send or receive, and wakes up after a 100 ms interval or when there is data to send, whichever event occurs first. During the period of sleep, if the access-point receives packets meant for the mobile device, it buffers the packets and advertises the presence of the buffered packets in periodic beacons to the mobile devices. A mobile device that wakes up after the 100 ms interval will poll the access-point for data if it finds that the beacons

advertise data buffered for that device.

When a mobile device is merely acting as a client (as in traditional client-server applications), the use of the PSM does not severely impact the initiation of connections because the radios are required to wake-up when there is data (or in this case TCP SYNs) to be transmitted, and clients always initiate connections. However, in a P2P network, where a mobile host can act as the server, when other peers initiate connections to the mobile host, the mobile host has no way of knowing that connection initiation messages are pending for reception at the access point till it wakes up from its fixed sleep interval, polls the access-point, receives all buffered data ahead of the initiation message one by one, and finally receives the initiation message.

Thus, the start-up time during connection initiation for the corresponding peers, when attempting to connect to a mobile peer that acts as a server can be strongly influenced by the PSM operations at the mobile host because of three reasons. First, each of the message/packet exchanges can be delayed in the form of inflated transaction time. Second, if the available bandwidths are low this problem is exacerbated because of the need to drain the buffered data at the access point that is queued ahead of the initiation message(s). Third, if the access-point PSM buffer is full when the initiation messages arrive, potential message drops and associated delays will arise as well.

Such adverse impact on the connection during its infancy also has other implications. For example, TCP's rtt and rtt_{avg} calculations are significantly impacted by even a small number of samples during connection initiation because of the small congestion windows during start-up. Figure 26(a) shows the impact of PSM on the performance experienced by peers trying to use the mobile peer as a server. We observe that turning on the PSM at the mobile peer acting as the server results in a throughput reduction of up to 40% depending on downloaded file size. Thus we see that the PSM can potentially harm the peer-to-peer connection. Figure 26(b) shows the actual time instants at which different TCP packets (both for BitTorrent protocol messages and actual data) are received when the PSM is either turned on or off for a single TCP connection. We observe that the TCP sequence number increases much faster when PSM is turned on, which proves that the impact on the

start up delay has a significant impact on the connection progress. In addition to Figure 5(b), we also provide the microscopic view of the start-up delay in Figure 5(c). The figure shows 6 BitTorrent protocol messages and the first data packet along the time line. We observe that the start-up delay of the data transmission is about 1 second.

3.4.4.2 Server Mobility

P2P peers update address list of other peers to establish connections to them. Peer address updates in BitTorrent typically happen at the granularity of minutes when a peer goes back to the tracker for an updated set of peers to use in its swarm. Even within that period, the decision on whether to use a peer or not for downloads is adjusted at the granularity of tens of seconds. Note that this is understandable in a wired environment where disconnections of peers may not occur frequently. However, when a mobile peer undergoes a hand-off or simply gets disconnected, the fixed peers who were clients with respect to the mobile peer will continue to try to reach the mobile peer till either the peer selection algorithm chooses an alternate peer (after 10-20 seconds) or the tracker provides alternate addresses or alternate peers (after a few minutes if not longer).

Figure 27 shows the effect of server side mobility on the throughput performance for fixed peers. The client is receiving data from three other peers (*i.e.*, the servers). The IP address of the server is changed every fixed time interval and the resultant throughput as enjoyed by the fixed peers is measured. We observe that as the time interval between

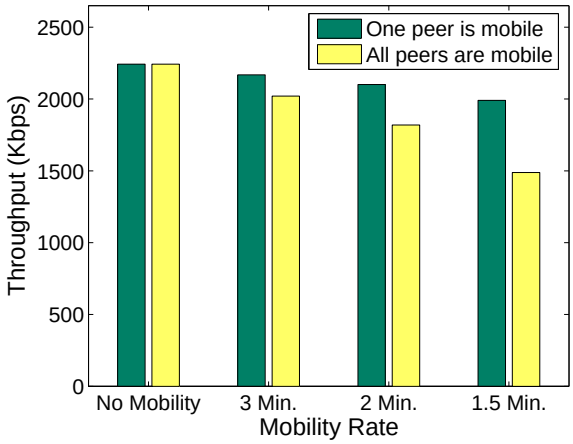


Figure 27: Impact of server mobility

successive IP change decreases (reflecting higher mobility) the throughput falls. Specifically, with current BitTorrent peer-updating model, the time required for a practical IP-address update of a serving neighbor takes the following three sequential steps: (i) the peer acquires a new IP address and reports to the tracker; (ii) the downloading peer requests for peer-list update from the tracker; and (iii) the downloading peer chooses the peer with a new IP and initiates the connection. These steps can take as much as several minutes depending on factors such as swarm size. Furthermore, the degradation due to mobility is amplified when the number of mobile peers amongst the corresponding peers is increased.

3.4.5 Relevance to Other P2P Networks

Thus far, we have investigated properties of the BitTorrent P2P data network that negatively impact performance of a mobile peer. While we use BitTorrent, a popular flavor of P2P systems, to describe the various challenges faced by P2P applications in wireless and mobile environments, we feel that it is necessary to broaden our research results to other P2P systems. Admittedly, a thorough investigation requires considerable research efforts and we view this as future work. Briefly, while not all the properties discussed thus far are directly relevant in the context of the other popular P2P data networks such as Gnutella, FastTrack, and eDonkey, many of the issues discussed in this work apply with varying extents. Specifically, (i) Any P2P network that uses TCP and allows data exchange on the same connection has issue of bi-directional TCP. We do notice that many current P2P softwares use TCP due to TCP's advantages when compared to UDP. (ii) P2P networks that accumulate incentives based on the amount of data transmitted to other peers has the upload-based incentive. For example, many flavors of Gnutella have incentive mechanisms. Such incentive mechanisms might suffer from mobility depending on the exact realizations. (iii) When media files that allow partial playback are shared by P2P networks, fetching sequences affect the playback performance. If non-sequential fetching algorithm is used, the playable percentage will be much smaller than the downloaded percentage. (iv) The proper functioning of P2P networks relies on the contributions of all peers. So when uploading data, server mobility and PSM on mobile hosts can have impact on the uploading

performance.

*3.5 w**P2P Design***

In this section, we first summarize the limitations with current P2P designs by characterizing the limitations into four problem classes. We then outline the key design aspects of the proposed *w*P2P solution that are targeted to address the limitations of existing P2P data networks identified in Section 3.4. We present the design basis for *w*P2P in the form of four principles that address the above four categories of problems. These principles are Role Reversal, Age-based Manipulation, Incentive Aware operations and Mobility-aware Fetching.

For each of the four principles, we first highlight the design rationales that serve as the basis of the each design principle. We then present the algorithmic details of the realizations of the different principles. We denote the realizations of the different design principles as *components*. We present the realizations in the context of a P2P client application, and hence, an implementation of the mechanisms would involve updating the P2P application. We believe that such a deployment model is very justifiable in the context of P2P data networks where it is not uncommon to receive updates to the P2P client application. Also, we present any specific implementation details with respect to the BitTorrent protocol. However, all the realizations presented are *purely local to the mobile host* and backward compatible to all existing BitTorrent P2P client applications on fixed peers.

3.5.1 Insights into the problems

Before we proceed to the design, we characterize the insights into the problems identified in Section 3.4 by categorizing the different limitations into four problem classes. Problems within a class all have a common underlying cause. We use the categorization to then position the design basis for *w*P2P. The limitations identified in Section 3.4 can be categorized as follows:

- *Mobile host as a server:* Unlike in other applications, in P2P data networks, the mobile host *uniquely functions as a server that can be accessed by other peers.* Hence, several problems attributed to wireless and mobile environments that can be (and have been) solved have to be solved differently. Moreover, some other problems related to such environments

that do not exist when the mobile host acts only as a client arise newly. Instantiations of this category of problems include the impact of PSM on the server performance, and server mobility.

• *Use of bi-directional TCP:* While bi-directional TCP has been in use by other applications such as http or secure-shell, *P2P data networks uniquely use bi-directional TCP to transfer large volumes data in both directions simultaneously.* Hence, several quirks of bi-directional TCP that do not arise in wireless environments otherwise have to be addressed. Instantiations of this category of problems include the ACK piggybacking problem and the DUPACK decoupling and overload during congestion.

• *Failure of incentives:* While P2P data networks heavily rely on incentives based mechanisms to encourage peers to contribute to the network, such mechanisms are not tailored for the unique characteristics of wireless and mobile environments. Specifically, the problems of upload-download self-contention and identity loss after mobility fall under this category of problems.

• *Disconnections and fetching:* The newest generation of P2P data networks use multiple simultaneous fetches from different peers and fetch blocks in a non-sequential order. While such strategies have obvious benefits in a wired environment, they compromise performance in a wireless setting. Problems that fall under this category include lack of playable content during disconnections and redundant fetching of partial blocks after reconnections.

3.5.2 Role Reversal

The problem of mobile host as a server primarily impacts connections either when the mobile host uses PSM or when the mobile host moves (and hence undergoes an IP address change). In this context, the *role reversal* design principle of *w*P2P involves the mobile host switching its role to that of a client at specific points in time to address the above problems. Note that the fact that a mobile host switches to acting as a client will not have any impact on the mobile host still serving content to the peers it connects to, as peers can serve traffic irrespective of whether or not they initiated the connection.

Essentially, for both the problems in this category, the TCP connection "suffers" due

to delays or disruptions, and the impact of the delays or disruptions can have a non-trivial impact even on the overall performance of the connection, especially when the file size or the remaining file size is small. Under role reversal, a mobile host acting as a server, upon detecting a connect request (*i.e.*, TCP SYN) delayed by the PSM or a hand-off, explicitly issues a TCP-Reset to the corresponding peer for previous connection. Further, the mobile host re-initiates a new connection to that same peer, thus acting as the client. Once the new connection is setup, because of the bi-directional nature of the application semantics and the bi-directional nature of the TCP connection, the mobile host can serve the content as a server. However, having replaced a "suffering" (i.e., the progress of the connection is unnecessarily delayed, in the case of PSM) or stalled (in the case of server mobility) connection with a newly formed "healthy" connection (i.e., the progress of the connection is smooth) allows the peers to achieve much better performance.

The Role Reversal (RR) component realizes this design principle and kicks in during connection setup and during mobility-induced network changes. Figure 28(a) shows the pseudo-code of the RR component. During connection initiation, the RR component captures packets and checks the TCP flag in the captured packet. If the SYN flag is set in the TCP packet and PSM mode is enabled on the wireless transceiver, then the RR component issues an *application-close* (Line 4 of Figure 28(a)) to the TCP layer triggering a TCP RST packet to the peer which initiated the connection through the SYN packet. After the application close, the RR component sends an *application-open* (Line 5 of Figure 28(a)) to establish a P2P connection to the corresponding peer. In this fashion, the RR component changes the mobile host to act as a TCP client as opposed to a TCP server. The RR component ignores the SYN packet if PSM mode is not turned on.

The RR component also monitors the IP address of the wireless interface. If it detects a change in the IP address, the RR component: (i) stores the necessary information of all the corresponding peers with which P2P TCP connections have been established, (ii) transmits application termination messages to all the stored peers, and (iii) issues application setup messages to the stored peers.

3.5.3 Age-based Manipulation

The bi-directional TCP problem arises because of specific quirks of the TCP design and how they relate to the wireless environment. However, at the same time, bi-directional TCP's performance otherwise is desirable since it eliminates ACK overheads under normal conditions. In other words, the solution to the problems with bi-directional TCP is not to switch back to dual uni-directional TCP connections, as doing so would render the overall performance worse than when using bi-directional TCP as pure ACKs in both directions and consume precious bandwidth resources.

In this context, the *age-based manipulation* design element of *w*P2P involves the adaptive manipulation of the bi-directional TCP connections for better performance. Essentially, recalling the discussion on ACK loss rates in Section 3.4, an argument can be made that TCP's throughput performance is vulnerable to ACK losses *only when the congestion window is small*. At larger congestion windows, the higher ACK loss rates do impact progress, but not significantly. Hence, under age-based manipulation, explicit conversion of piggybacked ACKs to pure ACKs is performed when the connection congestion window (*i.e.*, cwnd) is small. Note that although this manipulation is done at the receiver, standard techniques exist to track the sender congestion window at the receiver. Piggybacked ACKs are let through as-is when the congestion window is larger than a threshold. A straightforward value for the threshold is 6 as it can be shown that congestion windows less than 6 are highly vulnerable to losses in either direction [69].

Similarly, the use of pure ACKs during fast retransmit and the associated wireless link overload is addressed by throttling the number of packets down explicitly such that the total number of packets (including the pure DUPACKs) outstanding amounts to half the number of outstanding packets before the congestion detection. This is performed by explicitly dropping one-fourth of the DUPACKs in the reverse path during the first round-trip time after the first DUPACK is generated. For example, if the cwnd is 100 on congestion detection (assume one packet loss), 99 DUPACKs will be generated by the receiver. If one-fourth of the DUPACKs are dropped, approximately 24 DUPACKs will be dropped resulting in 75 DUPACKs reaching the sender. The sender, when it performs

fast retransmit and fast recovery, will thus send out 25 (*i.e.*, (100/2)+75-100) new packets. Thus, even if the 25 new packets generate 25 pure DUPACKs (assuming the receiver still is in loss recovery), that amounts to a total of 50 total packets equalling half of the number of packets outstanding before congestion detection.

Finally, the age-based manipulation principle also applies to one other technique of controlling when PSM is employed, and when it is not. Essentially, when a mobile peer has TCP connections in their infancy, PSM is turned off at the mobile. This includes the period right after role reversal is performed. The PSM is turned back on after the connections cross the infancy threshold in terms of their cwnd. Note that though there is an apparent performance tradeoff between the network performance (i.e., as measured by throughput) and power saving (i.e., as measured by operation time), in this work we explicitly focus on the network performance. We also note that it is possible to further study the throughput-vs-power tradeoff, as the qualitative impacts of PSM on throughput and power saving are not obvious, and we put this as future work.

The pseudo-code for the Age-based Manipulation (AM) component is listed in Figure 28(b). First the AM component constantly monitors the congestion window of the TCP connection. If the current connection congestion window is less than a specified threshold value γ (set to 6 in our evaluations [69]), the connection status is set to *YOUNG* (Line 3 in Figure 28(b)) and disables PSM on the wireless interface of the mobile host. If the congestion window is greater than the threshold γ, the connection status is set to *MATURE* (Line 6 in Figure 28(b)).

The AM component also maintains state about the TCP connection and captures TCP packets transmitted by the mobile host. If the connection status is *YOUNG*, the AM module conveys any new ACK information piggybacked on DATA packets transmitted by the mobile host as separate pure ACKs (Line 13 in Figure 28(b)). This achieves better robustness for the ACKs given a finite error rate in the wireless channel. The AM component also detects losses experienced by the TCP connection and performs the following operation during loss recovery. If the status of the connection encountering loss is *MATURE*, the AM module drops one DUPACK every four DUPACKs (Line 20). Specifically, *w*P2P captures and

manipulates TCP packets in *w*P2P using frameworks such as WinpkFilter [35] (for Windows OS) and NetFilter [21] (for Linux OS) that act transparently to the existing protocol stack of the network.

3.5.4 Incentive aware operations

The problem of failure of incentives stems from the two distinct conditions of the self contention in a wireless link and mobility-related identity loss. The *incentive-aware operations* principle in *w*P2P is used to address both problems.

Essentially, one technique under incentive aware operations in *w*P2P involves the adaptation of the upload rate in order to find the smallest upload rate possible to achieve the maximum download rate. While this value for the upload rate is trivial to determine in a wired setting, a more sophisticated algorithm is required in a wireless environment.

Since a wireless host uses a shared channel, the upload and download traffic contend with each other and hence increased uploads might reduce the downloads possible. In order to strike the balance between the two competing issues (*i.e.*, incentives and self-contention) *w*P2P performs a Linear Increase History-based Decrease (LIHD) algorithm that adapts the uploading rate. The intuition behind the LIHD algorithm is that while increasing the upload rate, it is better to be conservative so that the mobile host does not upload more than necessary. At the same time while reducing the uploads it is desirable to be aggressive, while at the same time performing as close to the optimal rate as possible. LIHD hence increases upload rates linearly when there is a positive correlation between the uploads and downloads, while decreasing the upload rates with increasing aggressiveness when decreasing the uploads does not cause a decrease in the downloads.

LIHD can also be used for controlling the rate of uploads when the mobile peer becomes a seed, such that the uploads do not impact negatively any of the downloads being performed by other non-P2P applications on the mobile peer. We do not consider this aspect of the mechanism in this chapter, and leave it for future work.

The Incentive Aware (IA) component monitors upload and download rates achieved by the P2P application and uses window-averaged throughput to determine the upload

rate control of the P2P application. Specifically, as shown in Figure 28(c), the component is invoked periodically. For each period, the downloading throughput is calculated and compared to the previous period. if the downloading throughput sees an increase, then it suggests that the uploading is unlikely interfering with the downloading and a larger uploading rate likely results in even better downloading throughput. Thus the uploading rate is increased linearly by a guard value of α (Line 6 in Figure 28(c)). Otherwise, if the downloading throughput decreases, then it suggests the opposite, and the uploading rate is curbed more aggressively. Specifically, the upload rate counter is decremented by a value proportional to the number of consecutive cutdowns of the upload rate (Line 8 in Figure 28(c))

IA also identities retention across hand-offs and within the same swarm. The rationale for generating uniquely different peer-ids in BitTorrent is to be able to identify and distinguish between clients with the same IP address (say, if the clients are behind a NAT), but at the same time confine the benefits of incentives accumulated by a peer to only that swarm in which the peer contributed. Since the typical scenario for task initiation in wired environments is when a peer wants to download another data file, generating a new peer-id is reasonable. However, in mobile environments task re-initiations can occur just because IP addresses have changed. *w*P2P, in this context, performs identity retention within a swarm, whereby even when task re-initiation is performed, as long as it is for a swarm the mobile peer was a member of before, the old peer-id is retained. This enables the mobile peer to leverage its previously accumulated incentives. The IA component stores the peer ID of the mobile host when the application is started and when there is IP layer handoff, the IA component restores the stored peer ID to maintain incentives.

3.5.5 Mobility-aware fetching

In Section 3.4 we observed how in a mobile peer (as a client) mobility can impact the performance of downloads in BitTorrent. In this context, *w*P2P uses a *mobility-aware fetching* principle that explicitly controls how data is fetched both during steady-state, and upon re-connections after disconnections.

The mechanism that falls under this principle is that of exponentially increasing altruism or exponentially decreasing selfishness. Essentially, a mobile peer fetches blocks in sequence with a probability p_s $(=1 - p_r)$, and fetches the rarest-first block with a probability p_r. During the initial phases of the download, the mobile peer uses a small value (say, 0.1) for p_r, and exponentially increases p_r as it downloads increasing fractions of the total file.

The rationale for this design is as follows: during the initial stages of downloads, if the mobile host gets disconnected, there is no benefit due to the rarest-fetch mechanism *either for the mobile host (in terms of playability) or to the P2P network (in terms of availability)*. Hence, it is more desirable to fetch sequentially. However, as the mobile host stays connected for a longer period of time, its utility to the P2P network has more stability and hence it is more meaningful to have available rare blocks. Furthermore, if the mobile host now gets disconnected, the user still has a considerable portion of in-sequence data for playback.

Mobility-aware Fetching (MF) component, as the final piece of the *w*P2P framework, helps in achieving ideal fetching of file sequences by being aware of the connection stability of the mobile host. The pseudo-code for the MF component is listed in Figure 28(d). The MF component performs rarest-first fetches with a specific probability P_r. This probability P_r is a function of the network stability of the mobile host as measured by the amount of time elapsed since the last network disconnection of the mobile host (or the start of the download). This mobility-aware adaptive content fetching achieves an optimal tradeoff between sequential content availability for disconnected usage of content and usability of content for other peers to download.

3.5.6 Integrated Operations

We can classify the operation of the four components of the *w*P2P framework with respect to the different periods of the P2P connection. The operation of the four components are illustrated in Figure 29. The Role Reversal component is employed during connection establishment and immediately after mobility-induced IP handoffs. The operations of the Incentive Aware component are performed during steady-state of the TCP connections.

Age-based Manipulation component kicks in during early stages of the connection and during congestion recovery periods. Finally, the Mobility-aware Fetching component is active during the steady-state period of the TCP connection and also after IP address change due to reconnection.

3.6 Performance Evaluation

In this section we evaluate the performance of wP2P.

3.6.1 Evaluation Methodology

The evaluation is performed both with NS2-based [114] simulations and prototype-based experiments. Since BitTorrent protocol in the study has various software implementations, and different implementations may differ in certain aspects of protocol standards, we believe such simulations with embedded design principles can help capture the trend of performance enhancement delivered by wP2P.

To further evaluate the effectiveness of our proposed wP2P solution, we built a prototype which implements most of the mechanisms contained in wP2P. The primary goal of building such a prototype is to prove that the proposed wP2P architecture does indeed work with real applications.

The metrics we study in this evaluation are average throughput, download size for a given time and percentage of playable content for media files. We compare our results with a default-P2P application under various scenarios.

3.6.1.1 Simulation Setup

We use the NS2-based simulator to evaluate all our algorithms. The topology used in our simulations is shown in Figure 31. The network consists of both wired and mobile hosts. All our algorithms are added as modules to the wireless nodes in the simulator. Unless otherwise specified, all the wireless links have a bandwidth of 2Mbps and use the IEEE 802.11b standard. The wired network has 50ms-delay links between the individual hosts. The mobile hosts $M1$ and some other peers contains the new wP2P implementation. The maximum number of upload connections from $M1$ is set to be 4, and the number of download

connections can range from 1 to 100. All traffic is generated using FTP over TCP, and data points in all results are averaged over 5 runs unless otherwise specified.

3.6.1.2 Prototype Setup

We use a prototype implementation of *w*P2P that is built as an enhancement to the CTorrent client version 1.2 [8]. CTorrent is a lightweight C++ implementation of BitTorrent protocol with about 10K lines of code. The four components of *w*P2P are implemented by either modifying the source code of CTorrent or adding a separate module which works with a packet filtering utility widely available in Linux distributions. Specifically, the Role Reversal, Incentive Aware Operations and Mobility-aware Fetching components are implemented by directly modifying the CTorrent source code, and the Age-based Manipulation component is realized with the assistance of Netfilter utility [21]. We now describe the network setup of our implementation followed by descriptions of the four individual *w*P2P components.

• *Implementation details.* (I) The prototype realizes RR by modifying the source code of CTorrent. IP address change induced by mobility will render all current connections unusable, and CTorrent client will eventually lose all live peers. Thus, the prototype stores the remote peers to which it is serving data, monitors the number of live peers, and infers mobility when there are no live peers existing. Once mobility is detected, the mobile host will immediately attempts to build new connections to the stored remote peers to resume serving data. (II) AM needs to determine connection ages, and the determination is based on the measurement of current congestion window. Since the information of congestion window typically is not available to the application itself, the realization of this component has to obtain such information from certain networking entities. Specifically, we choose Netfilter utility to assist the implementation of the component partly due to its wide deployment in Linux distributions. A module in the user space keeps track of the amount of data sent by the remote peer in every round trip time (*rtt*), and uses the current value as an estimate of that peer's TCP congestion window for the next *rtt*. We chose a congestion widow of size 9k bytes (approximately 6 full packets) as an indicator of the age of the flow. If the

window size is less than the threshold, the connection is considered to be young, and TCP ACKs are decoupled from the data packets. Otherwise, the connection is set as mature, and the packets are sent as such without modifications. (III) IA prototyping employs two techniques: identity retention and LIHD rate control. For identity retention a static peer ID is used in lieu of a randomly generated peer ID every time the IP address changes. For LIHD we modify CTorrent's in-built capability to control upload and download limits. The default CTorrent client allows uses to specify the upload and downloads limits. We modify it to allow the adaptive LIHD rate control algorithm described in the previous Section. We use bandwidth monitors to check the current upload and download rates for the algorithm. (IV) The basic CTorrent client does not implement the rarest-first fetching algorithm commonly used by BitTorrent clients. Hence we first implement the rarest-first fetching algorithm for the default client. We then modify this algorithm to prototype MF by including sequence information and awareness of download progress.

• *Prototype Testbed:* We use two wireless clients in a popular BitTorrent network to compare the performances of wP2P with a default version of BitTorrent. One of the clients has the modified CTorrent version and the other has a plain vanilla version of the CTorrent, which we refer to as default-P2P. Linux machines connected to the Internet through NS2 based wireless emulators are used for the two clients. An illustration of the network setup is shown in Figure 32. We use NS2 emulation [62] to study the impact of various issues of wireless environments like random wireless losses, mobility and bandwidth. Specifically, we emulate random wireless losses using random bit errors. We emulate mobility by changing the IP addresses of the clients using the "*ifup*/*ifdown*" commands in Linux. We also monitor the bandwidth consumed at each client to enforce bandwidth limitations.

(a) Role Reversal Component

```
1  Monitor packets received
2  If captured packet has TCP-SYN flag set
3      If PSM is turned on
4          Send Application_Close;
5          Send Application_Open to peer;
6      End If
7  End If
8  Monitor IP-addr of wireless NIC;
9  If IP-addr changes
10     Issue Application_Close to the peers;
11     Send Application_Fetch of partial blocks;
12 End If
```

(b) Age-based Manipulation Component

Variables

$CWND$: Congestion window of TCP connection
γ: Connection status threshold
$STATUS$: Status of the TCP connection
$DUPACK\_CNT$: Number of DUPACKs sent by MH

```
1  Calculate CWND of TCP connections;
2  If CWND < γ
3      Set STATUS to YOUNG;
4      Store PSM status and Turn PSM OFF;
5  Else
6      Set STATUS to MATURE;
7      Set PSM of wireless NIC to stored status;
8  End If
9  Capture TCP packet transmitted by MH;
10 If the packet is piggybacked ACK
11     If STATUS is YOUNG
12         Decouple ACK from DATA;
13         Send pure ACK;
14     End If
15 End If
16 If DUPACK is transmitted by MH
17     If STATUS is MATURE
18         DUPACK_CNT ++;
19         If DUPACK_CNT % 4 ==0
20             Drop DUPACK;
21         End If
22     End If
23 End If
```

(c) Incentive Aware Operations Component

Variables

U_{max}: Maximum upload limit
U_{cur}, U_{prev}: Current, previous upload state
D_{cur}, D_{prev}: Current, previous download state
α, β: Upload increment and decrement factor
U_{dec_cnt}: Upload decrement count

Initialization

```
1  U_cur = U_prev = 0.5*U_max;
2  D_cur = D_prev = 0; U_dec_cnt = 0;
```

Update

```
3  Determine current P2P download rate and store;
4  If D_prev <> 0
5      If D_prev < D_cur
6          U_cur = U_cur + α; U_dec_cnt = 0;
7      Else
8          Increment U_dec_cnt; U_cur = U_cur − β * U_dec_cnt;
9      End If
10 End If
```

(d) Mobility-aware Fetching Component

Variables

P_r: Probability of performing rarest-first fetch
σ: Probability increment value
Δ: Aging timer interval

Initialization

```
1  P_r = 0.1;
```

Maturity-timer Expired

```
2  P_r = P_r + σ;
3  Reset maturity-timer from Δ seconds;
4  Pick a random number r (0 < r < 1);
5  If r < P_r
6      Perform rarest-first fetch;
7  Else
8      Request the next in-sequence piece;
9  End If
```

Mobile Handoff

```
10 Set P_r = 0.1;
11 For each request stored
12 End For
```

Figure 28: Pseudo-code : (a) Role Reversal, (b) Age-based Manipulation, (c) Incentive Aware Operations, (d) Mobility-aware Fetching

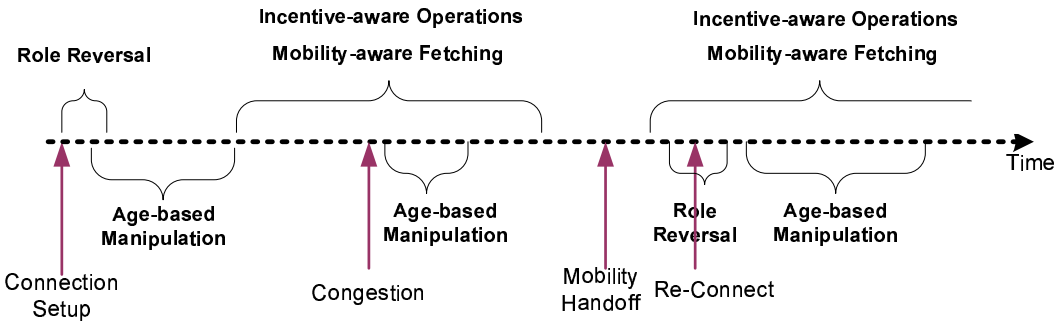
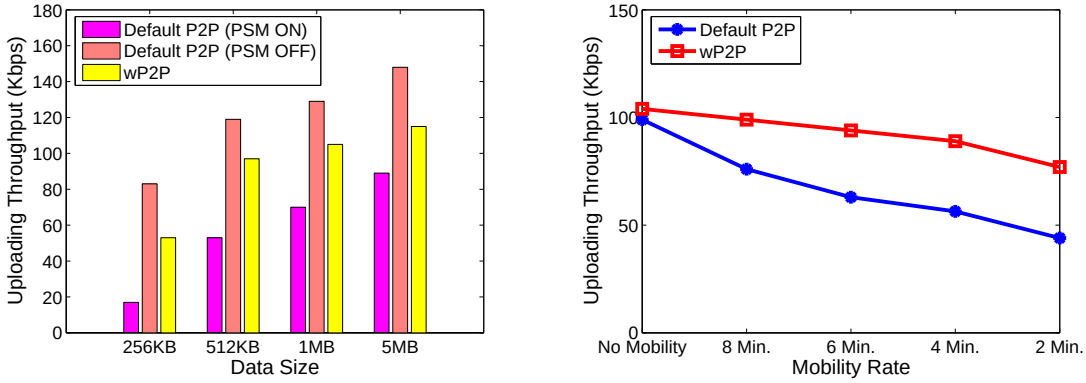


Figure 29: Integrated operations



(b) Impact of data size (No mobility) (a) Impact of server mobility

Figure 30: Role Reversal: Simulation results (a) and Prototype results (b)

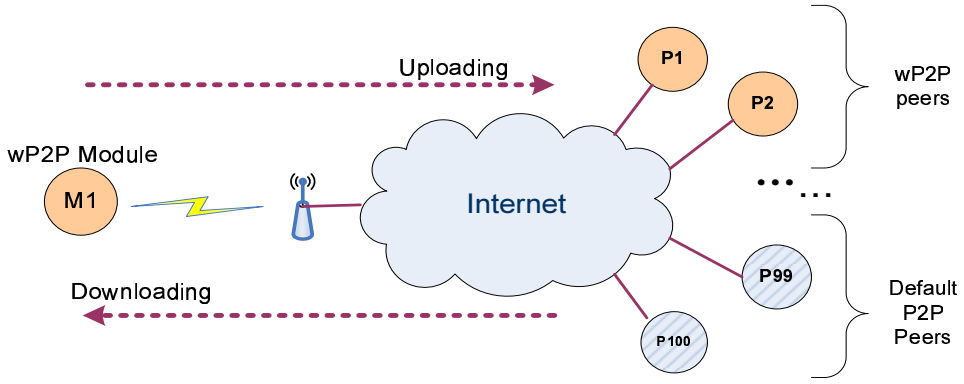


Figure 31: Simulation Setup

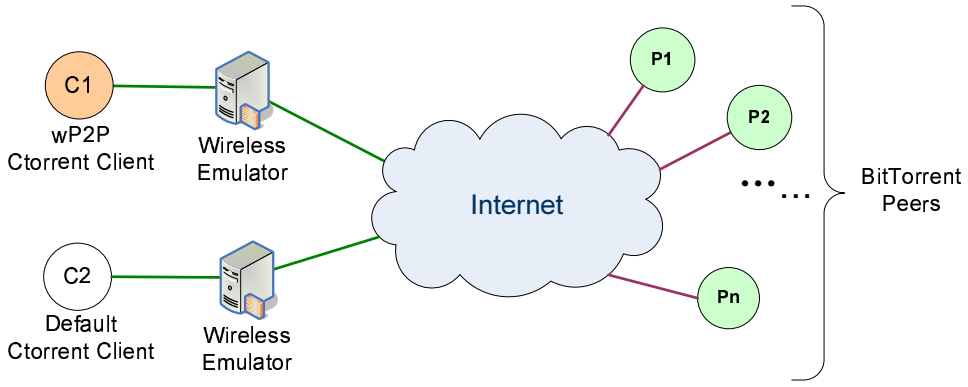
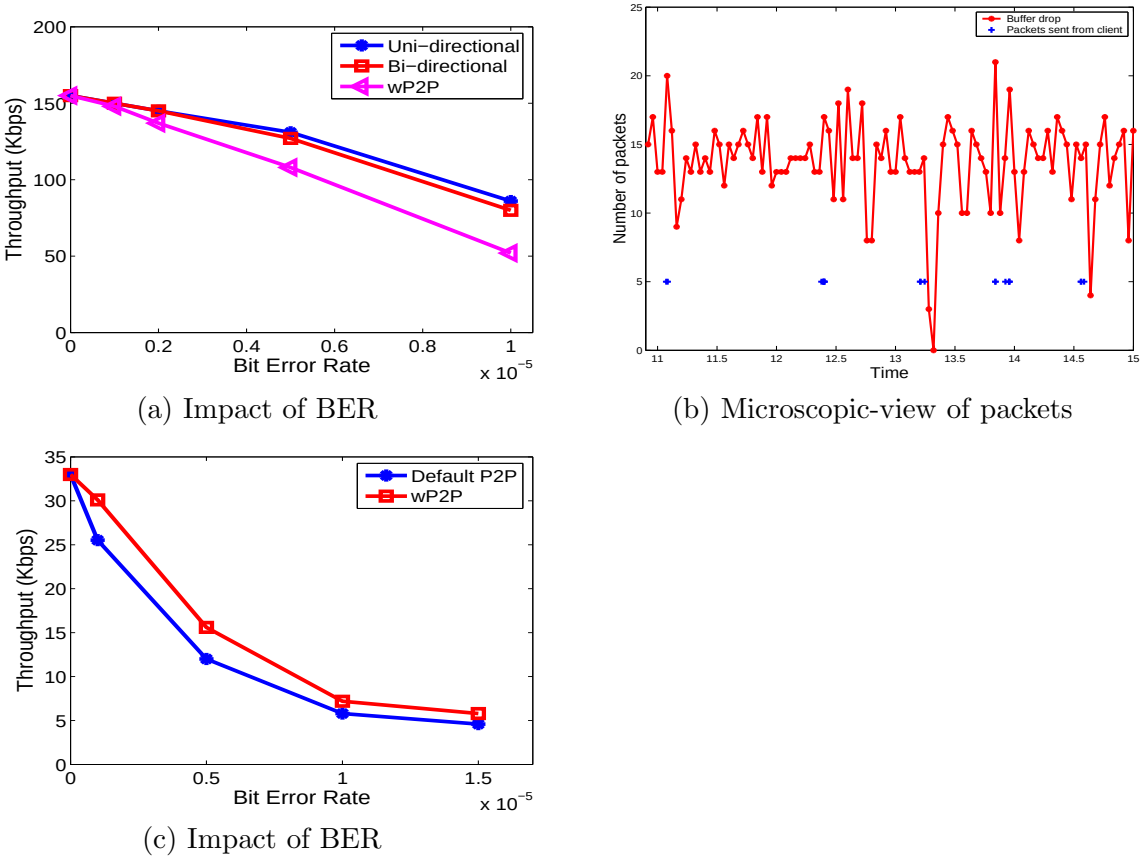


Figure 32: Testbed used in prototyping

3.6.2 Role Reversal

When a mobile host serves as a server, it receives requests from clients and sends data back. Figure 30(a) shows throughput obtained for varying file sizes at the server for default-P2P with PSM turned ON, default-P2P with PSM turned OFF and *w*P2P. The figure shows that PSM turned OFF gives the best performance. This is expected because when there is no PSM, packets can be sent as and when required. However, the presence of PSM affects default-P2P. This is because the startup delay for connection establishment increases initial *rtt* estimation, which results in lesser throughput. Using the role reversal component, a new connection is established with lower initial *rtt* thus improving throughput performance. Smaller file sizes show greater improvements because the effect of the start up delay is more significant.

• *Prototype Results.* To evaluate the role reversal technique we setup two mobile seeds in a live swarm which shares the file of Fedora-7-KDE-Live-i686.iso image [32]. Figure 30(b) shows the 10-run averaged uploading throughput of the two clients with different disconnection rates (*i.e.*, rate of change of the IP address). As the rate of connection disruptions increases the upload throughput naturally drops. The role reversal technique of *w*P2P allows the client to achieve far more upload rates when compared to the default client. We observe that with increasing rate of disruptions the performance improvement of *w*P2P is higher. This is because when there are frequent disruptions, the re-establishment time for default-P2P becomes a significant factor, whereas in *w*P2P the re-establishment



(a) Impact of BER

(b) Microscopic-view of packets

(c) Impact of BER

Figure 33: Age-based Manipulation: Simulation results (a,b) and Prototype results (c)

time is reduced. We observe improvements of up to 50% in wP2P when connections are disrupted every 2 minutes.

3.6.3 Age-based Manipulation

For bi-directional TCP, piggybacked ACKs are more susceptible to losses than individual ACKs. Figure 33(a) compares the downloading throughput observed using a uni-directional TCP connection, bi-directional TCP connection and wP2P as a function of bit error rate. The wP2P algorithm requires the receiver to know the sender's congestion window. In the simulation we assume the receiver side can obtain the current sender side congestion window size. Based on this value, the receiver opportunistically adapts the piggybacking of ACKs. In real life, TCP behavior inference tools such as TBIT [92] can be used to identify the sender side congestion window size at the receiver. With increasing bit error rates, the packet loss rates increase and results in smaller congestion windows. In this case

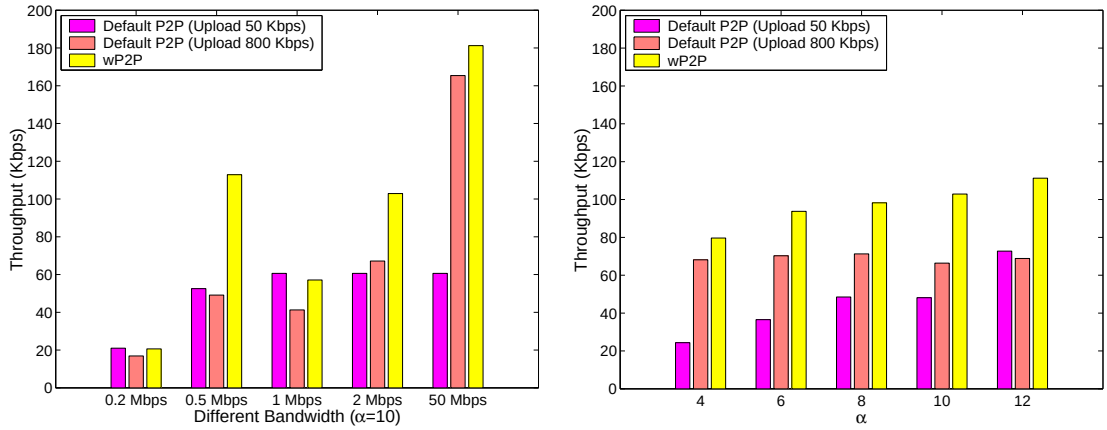
uni-directional TCP will give the best performance because ACKs suffer lesser losses than in the bi-directional case. *w*P2P performs very close to uni-directional performance.

In bi-directional TCP flows, we observe that the number of packets in transit originating from the mobile host does not reduce even when congestion occurs. Figure 33(b) shows the number of packets sent by the mobile host running *w*P2P at different times when there is a bi-directional TCP flow between the mobile host and a wired host. We also show the congestion events on the same graph. We observe that when a congestion event (*i.e.*, buffer drop at the AP) happens the number of packets in transit decreases for wP2P, albeit with a small time shift. This is because of the time taken for the information about the congestion event to reach the mobile host.

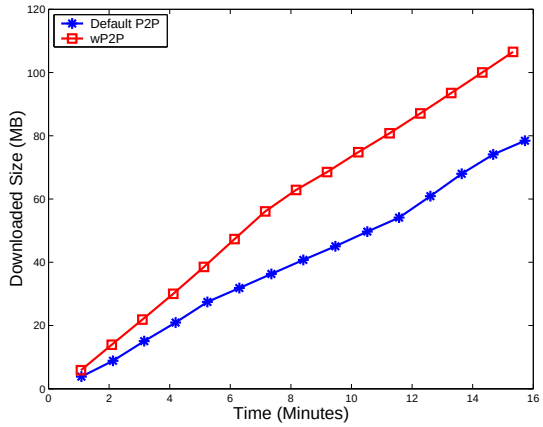
• *Prototype Results.* *w*P2P addresses the problems of Bi-directional TCP in wireless environments using the AM component. We study the impact of this component under varying random loss conditions emulated by varying the BERs ranging from 0 to $1.5e^{-5}$. We compare the download rates observed and show the averaged results in Figure 33(c). We observe that *w*P2P outperforms the default CTorrent under all bit error rates because decoupling ACKs result in smaller ACK losses for the connection, and in turn, larger throughput. Specifically, with all the four BER values, *w*P2P achieves about 15% to 20% more throughput.

3.6.4 Incentive Aware Operations

For evaluating the benefits of the LIHD algorithm in *w*P2P, we study the download throughput by varying two different parameters: α (*i.e.*, Upload increment) and *bottleneck_bandwidth*. We compare *w*P2P for two different upload rates, 50kbps and 800kbps, signifying low and high uploads. Figure 34(a) shows the effect of different bottleneck bandwidths. In the Figure, two default-P2P clients having fixed uploading rate are compared to *w*P2P. At low bottleneck bandwidths all three show similar download rates. This is because of the contention between uploads and downloads. However when the bottleneck bandwidths are higher, *w*P2P adapts to higher uploading rate, which leads to more downloads and thus *w*P2P achieves maximum download rate. In Figure 34(b) we vary the incentive parameter



(b) Impact of physical bandwidth (a) Impact of α



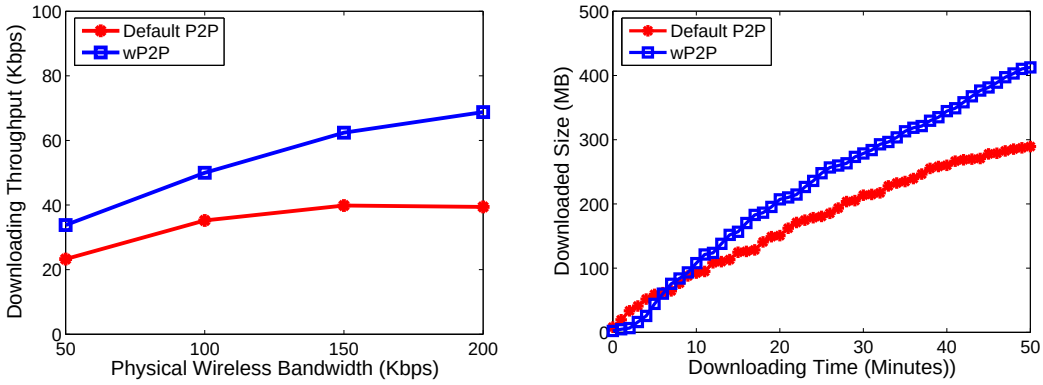
(c) Maintaining incentives

Figure 34: Incentive aware Operations: Simulation results

α. A large value of α results in higher downloads even for small uploads. wP2P would operate at the optimal upload rate that results in maximum download rates. We find that wP2P gives the best performance for all values of α. Figure

Figure 34(c) shows the total downloaded size as a function of time for default-P2P where incentives are lost during connection disruptions (due to mobility) and wP2P where incentives are maintained. We assume that the connection disrupts every one minute. For default-P2P whenever incentives are lost the download rate is reset to the minimum value unlike in wP2P where the incentives are maintained. Thus we find a larger total downloaded size for the same time using wP2P.

• *Prototype Results.* As discussed in the previous section identity retention and LIHD rate control address the issue of failure or loss of incentives. To evaluate the benefits of LIHD we vary the bandwidth of the wireless emulator from 50Kbps to 200Kbps. Figure



(a) Impact of physical bandwidth (b) Maintaining incentives

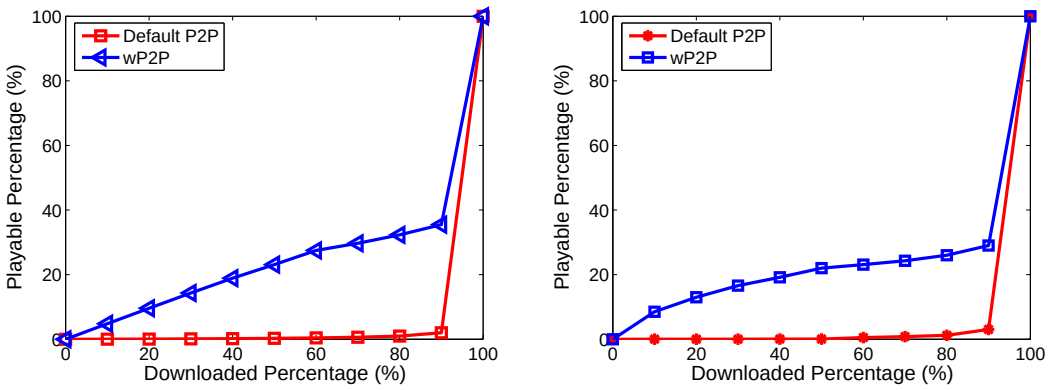
Figure 35: Incentive aware Operations: Prototype results

35(a) shows the averaged results over 10 runs for the case when $\alpha = \beta = 10Kbps$ (refer to Section 3.5 for definitions of α and β). We observe that, initially as the available bandwidth increases both wP2P and the default client show increased download throughput, but beyond a certain point the default client starts losing achieved throughput. With a bottleneck bandwidth of 200Kbps we observe that wP2P outperforms the default by as much as 70%.

To evaluate identity retention we use the two CTorrent clients to simultaneously download a Fedora-7-KDE-Live-i686.iso image [**?**], a 688MB file shared among more than two hundreds peers when our experiments were conducted. The IP addresses of the two clients are changed every one minute to emulate mobility. Figure 35(b) shows the total downloaded size of a typical run for these two peers. The downloaded size is plotted as a function of time. After 50 minutes of download we observe that wP2P downloaded about 100MB more than the default.

3.6.5 Mobility-Aware Fetching

Mobility-aware Fetching is capable of achieving two benefits from the client's perspective: enhanced playable percentage and improved downloading time. We show the effectiveness of adaptive fetching in Figures 36 (a) with respect to the playable percentage. The figure shows the result of a 100 MB file. We observe that adaptive fetching can achieve far better performance than default P2P fetching. For instance, when 50% of data has been downloaded, Adaptive fetching can achieve about 20% of playable percentage; on the contrary,



(a) Simulation results (b) Prototype results

Figure 36: Mobility-aware Fetching (100 MB files)

default-P2P fetching only attains 1.4%: a 14-fold difference.

• *Prototype Results.* Figures 36 (b) shows the results of the Mobility-aware Fetching of the content being downloaded and compare them against the default rarest first fetch algorithm. The results are averaged over 20 runs. In these experiments we set the value of p_r (*i.e.*, the rarest-first fetching probability) to be equal to the downloaded percentage of file. We observe that MF can achieve significantly better performance compared to the default P2P. For instance when 50% of the data has been downloaded, MF can result in about 20% of playable content while the default rarest first technique can achieve only about 0.5%.

3.7 Related Work

3.7.1 P2P Data Sharing Networks

Peer-to-peer data sharing networks have evolved in the way content searches and data transfer are performed. The first generation of P2P data sharing networks such as Usenet, Napster, etc performed a centralized search for content. The second generation of P2P networks like Gnutella [13] and FastTrack [10] use decentralized content searches. The third generation of P2P data networks (eDonkey [7], BitTorrent [3]) use both distributed searches and multiple downloads from several peers to accelerate the content fetching process.

3.7.2 P2P Enhancements

There are many works in literature that propose to improve performance of P2P systems. Some of these works focus on incorporating better incentive schemes to encourage cooperative behavior and penalize free riders. Reputation-based trust systems ([59, 81, 112]) and key sharing protocol ([123]) are works in this category and these work try to prevent non-contributing nodes from gaining undeserved benefits from the system. Other work ([85, 118]) design mechanisms to generate unique peer-IDs that feature desired properties. Some work ([46, 79, 95, 98]) analyze the performance characteristics of the BitTorrent protocol. In comparison with these works, our work is focused on the unique challenges arise as mobile hosts join the p2p networks. These challenges are not seen in fixed hosts and in wired networks. For instance, we deal with the incentive loss problem experienced by mobile hosts instead of proposing a new incentive mechanism.

Recently more and more p2p users go mobile and are connected with wireless links. The authors in [67] analyze the traffic pattern of a well-established 802.11 WLAN network and show that P2P traffic including P2P data sharing and streaming has increased dramatically. The authors in [55] propose a cross-layer optimization of Gnutella for deployment in purely mobile ad hoc networks. [71] designs an algorithm to select a new resource provider for mobile peers when mobility occurs to the remote peers. Our work on the other hand looks at the effect of mobile peers on an existing P2P network.

3.7.3 Mobility

Works abound to address the mobility issue. Mobile IP (RFC 2002) [97] is the current IETF standard for supporting mobility on the Internet. It provides transparent support for host mobility by inserting a level of indirection into the routing architecture. Work [109] deals with end-to-end applications and designs an architecture for Internet host mobility. It uses dynamic updates to the Domain Name System to track host location. Work [122] addresses the issue of mobility in an ongoing transport connection by providing transparent network connection mobility using reliable sockets (rocks) and reliable packets (racks). In work [60] a mobility-aware file system for partially connected operation is presented. Specifically, it allows applications to maintain consistency on only the critical portions of its data files. In work [71] the authors design an algorithm to select a new resource provider for mobile peers when mobility disconnects remote peers. In comparison, our work looks at issues faced by generic P2P data sharing applications on mobile hosts, identifying a variety of challenges that arise when mobile hosts join the P2P networks and act as P2P server.

3.7.4 PSM

Power save mode in 802.11 is proposed to save energy on mobile clients. To maximize the efficiency of PSM, work in [77] proposes BSD (bounded slowdown) to adapt sleeping durations depending on past activities to ensure no connections are punished more than a factor of p. Work in [39] suggests a set of design elements to turn on or off power save mode according to applications. Work in [124] introduces proxies to buffer data in order and proposes a new scheduling algorithm to decide which flow should be served at which time, thus mobile clients can sleep as long as possible. Similarly [52] performs traffic shaping for applications and help the prediction approach more efficient. Work [120] adjusts the waking up and sleeping time of the NIC based on prediction of the next incoming packets. In streaming systems, this approach is less effective because data are typically received continuously. One solution is to transmit data packets as bursts, which allows NIC to sleep more time between bursts. However the bursty-ness may cause unnecessary congestion. Work [76] studies these impacts and adapt burst length to achieve improved

trade-off between power efficiency and congestion tolerance.

3.8 Conclusions and Future Work

In this chapter we have investigated the issues with using mobile hosts as peers in the P2P network. We identify several insights into the issues such hosts face using a real-life BitTorrent P2P data network. We then propose a solution called *wP2P* that significantly improves performance.

Open research issues include the exact translation of the solutions to other P2P data networks, and consideration of the issues and solutions for other wireless network environments.

CHAPTER IV

IMPROVING ENERGY EFFICIENCY OF LOCATION-BASED APPLICATIONS ON SMARTPHONES

4.1 Summary

Over the years, location-based applications (LBAs) have become increasingly popular. The usage of these applications, however, can cause severe battery drain in mobile devices owing to their power-intensive location-sensing operations. This chapter presents an adaptive location-sensing framework that significantly improves the energy efficiency of smartphones when running LBAs. The underlying design principles of the proposed framework involve suppression, substitution, piggybacking, and adaptation of applications' location-sensing requests to conserve energy. We have implemented these design principles on Android-based smartphones as a middleware. Our evaluation results on our implementation show that the design principles reduce the usage of the power-intensive GPS (Global Positioning System) by up to 98%, and improve battery life by up to 75%.

4.2 Introduction

With smartphones becoming increasingly pervasive over the past years, many Location-Based Applications (LBAs) have been adopted by mobile users for always-on contact such as social-networking, businesses needs, and entertainment. Some instances of popular LBAs include mobile social networking (e.g.,Twitter, FaceBook [9, 12, 33, 34]), healthcare (e.g., HealthMate [1]), local traffic (e.g., [27,68,72,88,121]), and local restaurants (e.g., OpenTable [24]).

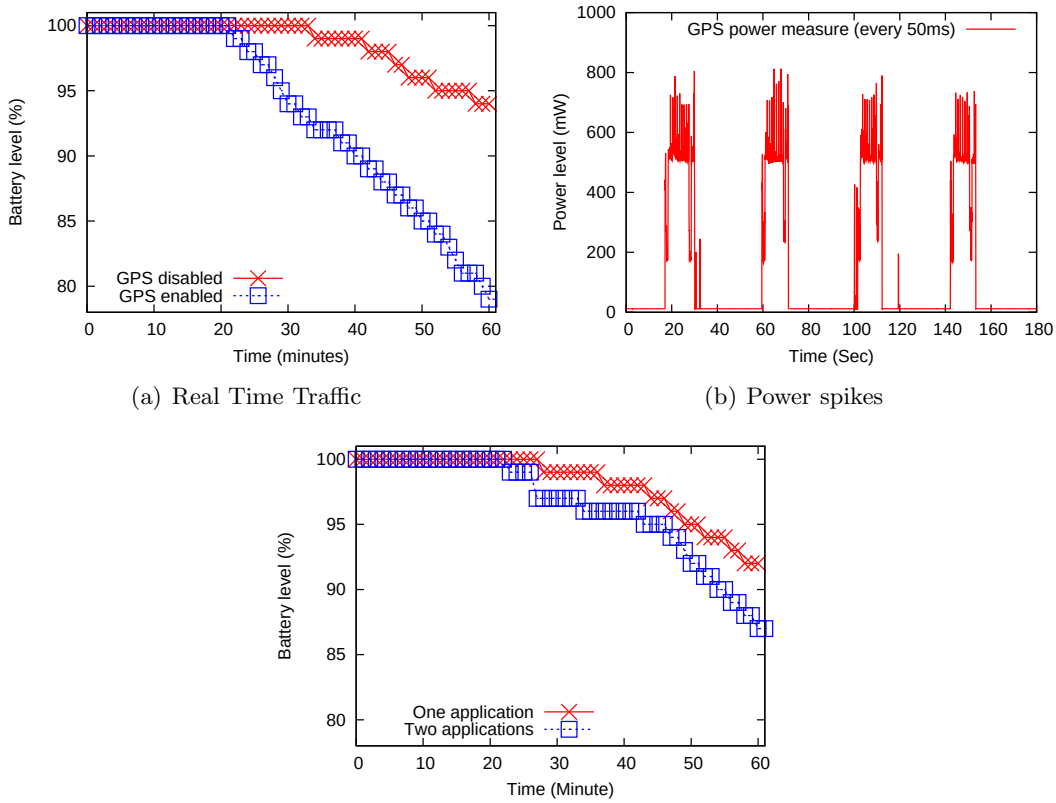
In spite of increase in processing power, feature-set, and sensing capabilities, the smartphones continue to suffer from battery life limitations, which hinders the active usage of LBAs. Typical battery capacity of smartphones today is barely above 1000mAh (e.g., the lithium-ion battery of HTC Dream smartphones has the capacity of 1150 mAh). Unfortunately, GPS (Global Positioning System), the core enabler of LBAs, is power-intensive, and

its aggressive usage can cause the complete drain of the battery within a few hours [45, 54]. Though the aggressiveness of GPS usage depends on specific applications, several flavors of LBAs such as local traffic (e.g., [27]) and social networking (e.g., [33]) particularly benefit from the continuous location updates. Real Time Traffic [27], for instance, requires continuous GPS location updates. Twidroid [33], a mobile version of Twitter, features a GPS accuracy booster, which provides the option to enable/disable continuous GPS sensing.

Numerous solutions have been proposed to improve the battery life of mobile devices [40, 107, 110, 115], but little attention has been given in the context of LBAs, simply relying on the intelligence of application developers. The LBA developers are suggested to reduce the use of GPS by increasing location-update intervals (say, to more than a minute), thus allowing GPS hardware to sleep between successive location-updates. Such a simple solution can improve battery life by forcing applications to request less frequent location information, but it has fundamental limitations. For instance, although reducing GPS invocation frequencies of each LBA saves energy, the effectiveness of this approach could be compromised when multiple LBAs are running, as the asynchronous use of GPS from different LBAs unnecessarily leads to an increased number of invocations.

In this chapter, we present an energy-efficient location-sensing framework that effectively conserves energy for smartphones running LBAs. In its core, the proposed framework includes four design principles: Suppression, Piggybacking, Substitution, and Adaptation. Briefly, *Suppression* uses other less power-intensive sensors such as accelerometer to suppress unnecessary GPS sensing, when the user is in static state. *Piggybacking* synchronizes the location sensing requests from multiple running LBAs. *Substitution* makes use of another location-sensing mechanism (e.g., network-based location sensing) that consumes lower power than GPS does. *Adaptation* aggressively adjusts system-wide sensing parameters such as time and distance, when battery level is low.

We have implemented the four design principles on G1 Android Developer Phone (ADPs) as a middleware and have evaluated the implementation extensively via measurements. While the proposed design principles are general enough to be applied to any layer, the



(a) Real Time Traffic

(b) Power spikes

(c) Multiple LBAs

Figure 37: Energy Consumption of Gps

middleware approach allows for better application transparency in the sense that applications can be kept as-is. We choose Android-based smartphones for prototyping, mainly because of its openness and predicated popularity in smartphone markets [23]. Our evaluation results with the implementation show that the proposed framework significantly saves energy in location sensing. For instance, in various scenarios, our prototype reduces the number of GPS invocations by up to 98%, and thus improves the battery life by up to 75%.

To summarize, this work makes the following contributions:

- We introduce an energy efficiency issue in location sensing for smartphones running multiple LBAs. To our best knowledge, this present work is the first to consider the energy issues under multiple LBAs environments on smartphones.
- We present four design principles that reduce energy consumption on location-sensing for resource-constrained smartphones. We further show that the integration of the proposed design principles provides significant amount of energy saving.

- We prototype the proposed design in Android-based smartphones, which are open to both practice and research, and demonstrate its effectiveness through real-life measurements.

The remainder of the chapter is organized as follows. Section 4.3 describes the motivation of this chapter. Section 4.4 presents the key design principles as well as their integrated operations. Section 4.5 describes our implementation. Section 4.6 shows evaluation results of our prototype. Section 4.7 discusses related work, and finally Section 4.8 concludes this chapter.

4.3 Motivation

In this section, we highlight the motivation of this work by presenting a set of experimental results. We first show the impact of several factors on the energy efficiency in location sensing using G1 ADP phones. Then, we summarize the limitations of existing smartphone usage with respect to energy-efficient location-sensing.

4.3.1 GPS Energy Consumption

We first assess the impact of using power-intensive GPS on smartphones. We consider a scenario where a user is driving with a traffic-monitoring LBA called "Real Time Traffic" running. The application is popularly used to determine traffic speed on the road network based on anonymous collection of users' locations, speed, and direction information [27]. While running this LBA (version 1.0.2e(17)), we measure instantaneous battery levels of the phone over an hour, using power-APIs provided by Android Software Development Kit (SDK).[1] For comparison, we also run the same LBA on the second phone with GPS disabled. For both experiments, we start with a fully charged battery after charging for same amount of time. The screens of the phones are always kept on. The map refreshing rate and GPS invocation interval of the LBAs are set to 5 seconds.

We use two brand-new ADP phones to perform experiments [2]. Figure 37(a) shows the

[1]Though the power-APIs of Android SDK only provide coarse-grained measurement of battery levels, we use them to show macro-scale impact, which is an interesting factor in this work.

[2]We also switch the phones to accommodate possible battery difference, and we observe similar results in our experiments. So in later presentation, we will only show the one run of results for simplicity.

instantaneous battery level of the phones during the run. As shown in the figure, when GPS is used, the battery level drops to 79% within one hour, whereas the battery with disabled GPS drops to only 94%. Note that although we run the experiment multiple times with different setups such as charging time, we employ no aggregated metric as each result varies, depending on the uncertainty in the battery mechanics (e.g., how many times the battery has been charged). However, we always see the same trends in battery drops across all runs.

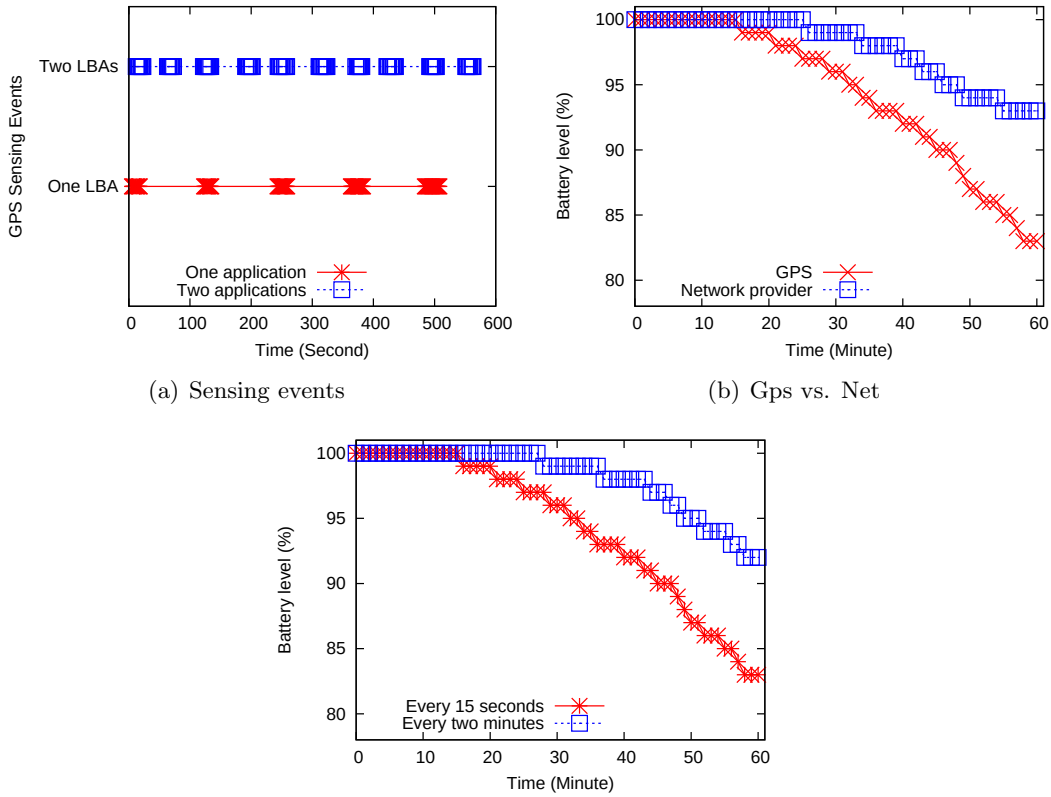
We also measure instantaneous power-spikes of GPS sensing using a digital multi-meter (Agilent 34410A) to see microscopic power usage. Figure 37(b) shows the power spikes of the phone (measured once every 50 ms) when running a LBA requesting GPS. As shown in the figure, a typical GPS invocation consists of a locking period and a sensing/reporting period. The lengths of these two periods are about 4-5 seconds and 10-12 seconds, respectively. More importantly, GPS sensing consumes about 600mW of power. For a typical battery capacity of 1000mAh such high power consumption is very expensive as continuous GPS sensing can deplete the battery in merely 6 hours (i.e., $\frac{1000mAh*3.7V}{600mW}$).

4.3.2 Multiple Location-Based Applications

GPS power consumption might become even more significant if multiple LBAs are running simultaneously.[3] Let us consider the following scenario. A user is initially running a social network LBA such as FaceBook on his Android phone and continuously publishing his locations. After a while, he begins to drive and launches a traffic-monitoring LBA such as "Real Time Traffic". Now both LBAs are running concurrently. Assuming both applications invoke GPS sensing every 2 minutes (i.e., with 2-minute invocation interval), GPS ideally needs to wake up every *two* minutes. However, if these two applications are not synchronized on GPS sensing requests, then GPS might need to wake up every *one* minute.

In Figure 37(c), we show the impact of multiple LBAs with two scenarios. In the first scenario ('One application'), only one LBA is running and requests GPS sensing every 2

[3]Smartphones such as those based on Android or Symbian support multitasking. The background LBAs still triggers location sensing.



(a) Sensing events

(b) Gps vs. Net

(c) Sensing intervals

Figure 38: Energy consumption of Gps and Net

minutes. In the second scenario ('Two applications'), two such LBAs are running without synchronizing GPS sensing request. As shown in the figure, when one LBA is running, the battery level drops to about 92% after 1 hour. With two LBAs running, however, the battery level drops to 87%.

To better understand the results, we plot the GPS invocation events during the first 10 minutes for both scenarios in Figure 38(a). As shown in the figure, when the sensing events of multiple applications are not synchronized, the GPS is indeed totally invoked 10 times rather than the desired 5 times, thus causing more energy consumption than when multiple LBAs are synchronized.

4.3.3 Multiple Sensing Mechanisms

Today's smartphones typically support multiple location-sensing mechanisms (or location providers). Android, for example, supports two mechanisms: GPS and Network-based triangulation. Network-based mechanism collects information about reachable cell towers

(or WiFi access points) from a mobile device and determines location from a location database. For simplicity, in the following presentation, we use 'Gps' and 'Net' to refer to these two location-sensing mechanisms, respectively. For clarification, we use GPS to refer to the physical device of Global Positioning System.

These two mechanisms have different accuracy and power consumption levels. In Figure 38(b), we show the power consumption of each mechanism, as one LBA is running with a location sensing interval of 15 seconds. The Net mechanism uses GSM cell towers to determine locations as both WiFi and 3G are turned off for both experiments. Net only causes the battery to drop to about 93% and consumes much less power than Gps does (i.e., 83%).

We also perform experiments to show the two mechanisms' accuracy. As shown in several prior studies, Gps can achieve an accuracy of as high as 10m in outdoor areas, while Net's accuracy varies depending on environments. To further understand such characteristics, we also perform experiments to measure Net's accuracy as follows. Due to the lack of a more accurate measure, we use Gps as ground truth to measure the accuracy of the Net. We perform experiments in an urban area around Silicon Valley, California. Net accuracy is measured as the average distance between the Gps-reported location and Net-reported locations. We observe that Net achieves an accuracy of about 30 meters to 100 meters during most of the time. Although Net still provides much coarser accuracy than Gps does, such accuracy might be sufficient for many LBAs (e.g., weather information).

4.3.4 Sensing Intervals

For many mobile platforms including Android, applications are allowed to explicitly specify the location sensing granularity in terms of updating time interval and distance interval. Intuitively, larger time and distance intervals can help save energy. In some scenarios, particularly when the battery level is low, LBAs can cooperate by explicitly increasing location-sensing intervals of time and distance (e.g. updating every 1 minute or 20 meters rather than every 30 seconds or 10 meters). To study the impact of adapting sensing intervals, we consider two LBAs with GPS invocation intervals of 15-second and 2-minute,

respectively. Figure 38(c) shows the progression of battery level. As shown in the figure, by simply enlarging the update interval from 15 seconds to 2 minutes, the application can help conserve 9% of battery level in one hour.

4.3.5 Problem Characterization

Figure 39 summarizes the problems identified above and additional intuitions in the energy efficiency of location sensing.

- *Static use of location sensing mechanisms:* In many cases, mobile platforms lack the dynamic selection of location sensing mechanisms. Many smartphones today support two major types of location-sensing mechanisms—Gps and Net. These sensing mechanisms have performance tradeoffs in terms of accuracy, power consumption, and dynamics. However, mobile platform statically uses its sensing mechanism, and this static use can lead to energy inefficiency in many scenarios.

- *Non-use of power-efficient sensors to optimize location-sensing:* Depending on specific environments (e.g., inside buildings) or contexts (e.g., phones being static), certain location-sensing operations may be impossible or unnecessary to perform, and thus blindly requesting location sensing wastes power. The environment and context information, interestingly, can be obtained by using other types of sensors that are more power-efficient. Many smartphones are typically equipped with multiple sensors such as accelerometer and orientation sensors, which consume much less power than those used for location sensing. Therefore, leveraging these sensors can optimize location sensing and conserve energy.

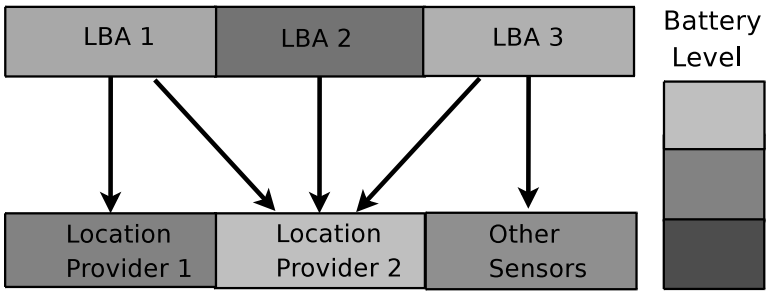
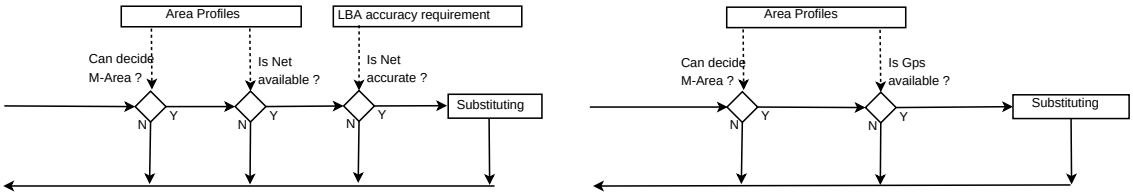


Figure 39: Problem characterization



(a) LBA requesting Gps (b) LBA requesting Net

Figure 40: Sensing Substitution

- *Lack of sensing cooperation among multiple LBAs:* When multiple LBAs run and request location sensing independently, they are not aware of each other, and their location-sensing operations are not coordinated. This results in *redundant* location sensing invocations and causes unnecessary energy consumption.

- *Unawareness of battery level:* When the battery power level is low, users are usually willing to tolerate degradation of location-accuracy or, at least, seek such an option in favor of longer operation time. Current mobile platforms, including Android, typically lack advanced battery-aware location management to strike a balance between location sensing accuracy and operation life.

4.4 Design

To overcome the limitations characterized in the previous section, we present four design principles and their integrated operations in a smartphone. Furthermore, we discuss performance tradeoffs in employing these design principles.

4.4.1 Sensing Substitution (SS)

Current smartphones lack the capability of selecting the most appropriate location sensing mechanism on-the-fly to strike the performance balances amongst energy consumption, availability and accuracy. LBAs are allowed to choose location-sensing mechanisms at the moment when they register their location-sensing requests to underlying systems. For instance, current Android SDK 1.5 allows an application to specify criteria indicating the applications' requirement about accuracy, power consumption, bearing (e.g. direction) and speed. Based on such criterion, the underlying framework chooses the most appropriate

mechanism (e.g. Gps or Net). Thereafter, the chosen mechanism will be always invoked, irrespective the changing environments or contexts.

Lack of dynamic selection of location sensing mechanisms leads to energy inefficiency as well as failure in satisfying LBA requirements. For example, in certain indoor environments and dense urban areas, Gps may not be able to provide accurate location information. Similarly, the performance of Net is heavily affected by the environment. For instance, in certain urban areas, studies have shown that Net can achieve as much accuracy as Gps does. On the other hand, in rural areas with a few cell tower available, Net shows low accuracy. Thus, with static selection of location sensing mechanisms, applications may not be able to effectively function, especially when the user moves around with LBAs running. For example, when Gps is used, applications expect to receive accurate location information all the time. However, if the environment prevents Gps from working, continuously invoking GPS apparently is wasteful in terms of battery energy. The same is true for using Net.

Our solution to these problems is a dynamic selection approach which we refer to as "Sensing Substitution (SS)". SS can choose the most appropriate location sensing mechanism on-the-fly. Specifically, SS is context-aware and can learn the characteristics of the location providers along the routes where phones move. It then performs location sensing in a more energy efficient manner by choosing the best sensing mechanism, given the context. Because typical mobile users routinely follow certain routes (e.g., commuting between offices and home) and visit familiar locations (e.g., restaurants, malls), and because these places exhibit consistent location-sensing related environment characteristics, such as GPS, signal strength and the number of APs, utilizing the environmental information can assist in choosing the most appropriate location provider.

To achieve dynamic selection of location providers, SS relies on learning environmental characteristics such as the availability and accuracy of location providers (e.g., Gps and Net). For this reason, the design of SS includes a location-sensing characteristic profiler. The profiler monitors and stores relevant information, including current locations, visit frequency, and sensing characteristics (e.g., availability, positioning accuracy) of location providers. The profiled data consists of a list of entries, and each entry corresponds to a

profiled area which we refer to as *M-Area*. M-Areas represents physical areas with geographical boundaries. In particular, the locations in the same area exhibit similar location-sensing characteristics. We will detail the rationale and data structure of M-Area in Section 4.5.7.

With the profiled areas, SS dynamically decides an optimal location-sensing mechanism as follows. For ease of illustration, we consider Android platform and show the high-level operations of SS as shown in Figure 40. Specifically, let's assume that the currently registered location-sensing mechanism is Gps. When the user moves into an area where Net is available and its accuracy can fulfill the LBA's requirement, then the LBA uses Net to replace Gps. As shown in Figure 40(a), SS first attempts to decide the most appropriate M-Area. Then, it checks Net's availability and accuracy. If Net's accuracy can satisfy the requirement of the LBA, SS performs substitution. Similarly, as shown in Figure 40(b), when the current location-sensing mechanism is Net and the phone moves into areas where Net is not working, SS invokes Gps, instead of Net. Since GPS consumes more power, SS uses less frequent GPS sensing to maintain the same level of energy consumption.

The characteristic profiler can be designed to be automatically obtaining profiling results including physical locations and availability/accuracy of location providers. To ensure higher degree of accuracy and energy efficiency, the profiler design also includes the following mechanisms. (i) The profiler may also involve users to explicitly control the profiling process. For example, users may specify the area boundaries of the profiler. (ii) The profiler calibrates either periodically or conditionally, depending on the changes in the profile characteristics. Essentially, whenever there is need to run profiling, the process will be invoked on demand. For instance, when the user moves to a new city to join a new job, the profiler will detect that change and proactively initialize the profiling process to accommodate the environmental change. In particular, when the profiler is first initialized, it performs profiling. After that, the profiling process keeps monitoring the necessity of performing profiling again. The necessity is measured by an opportunistic verification process. Specifically, it is periodically invoked to measure the location-sensing characteristics and compare them with the information stored in the profiler database. If the comparison results in a large discrepancy value, it indicates that another profiling is needed. The periodical verifications are

piggybacked on the existing location-sensing requests, and thus they do not incur additional sensing requests.

4.4.2 Sensing suppRession (SR)

Smartphone users may use the phones in various scenarios, and continuous location sensing may often not be needed. For instance, when the smartphone is in static state such as being put on a table in an office, continuous location sensing is unnecessary. It is desirable to "suppress" the sensing from energy efficiency standpoint. The design principle of Sensing suppRession (SR) is to detect phones' mobility state by using less-power-intensive sensors and to suppress unnecessary invocation of location sensing. The basic mobility-state information is whether the phone is static or moving, but it can contain more sophisticated information such as moving speed and direction.

The fundamental requirement of this design principle is to learn the mobility state (e.g., static or moving) of a phone with energy-efficient sensors. There are many existing research efforts ([117], [84], [75]) that attempt to profile users' mobility pattern. For example, SoundSense [84] uses the microphone to determine the user's logical location. In this work, we are primarily interested to use *low-power* sensors to suppress *high-power* location sensing. Specifically, we attempt to use sensors such as accelerometer and orientation sensors to profile smartphones' states. Other sensors such as camera and microphone used by the mentioned works, typically consume much more power than the low-power sensors, and thus are not considered in this work.

A challenge that arises is to ensure the correctness of mobility state detection. False positives of the extraction (i.e., falsely detecting that the phone is moving while it is not) will lead to the unnecessary location sensing, while false negatives will cause more serious consequences on LBA performance for changing locations. We propose various methodologies to reducing these errors, particularly the false negatives. First, a configuration option is exposed to a user, allowing the user to manually enable/disable a suppression option. Second, the aggressiveness of suppression is automatically adjusted based on information such as the confidence levels of the learned mobility context. The confidence levels reflect

the familiarity with the current mobility contexts such as commuting routes. Third, suppression is adjusted based on the application requirements. For example, if the application requires very coarse-grained location information, the suppression will be invoked. Fourth, a verification mechanism is employed to verify the correctness of the detection. Briefly, location sensing is periodically invoked for verification purpose even in a suppression mode.

4.4.3 Sensing Piggybacking (SP)

Sensing Piggybacking (SP) is designed to improve the energy efficiency of location sensing when multiple LBAs are concurrently running. It can re-use the existing sensing registrations by piggybacking new sensing requests on existing ones, thus eliminating some location-sensing invocations. For example, let us assume that a existing LBA registers GPS location-sensing every 2 minutes. When a new LBA starts and requests Gps with the same time interval, it can simply piggyback on the existing registered requests, thus avoiding separate sensings. Reducing the number of separate sensing can help save the energy associated with sensing as the sensing hardware can go to sleep between consecutive invocations.

Applications may request and register location sensing in various ways, as supported by the underlying framework or system. Android platform, for example, allows application designers to perform two types of sensing registration. In the first type, the application statically registers a location listener to the underlying framework, and the framework will periodically notify the listener of location updates based on the specified parameters such as time interval and distance interval. This method is simple, but it relies on the underlying framework for GPS to sleep between two sensing invocations. For example, if a Gps request takes 30 seconds to perform one invocation of sensing and if the specified time interval is more than 30 seconds, then the framework can turn off the GPS and put it into sleep to conserve energy.

The other type of registration is to explicitly register/unregister GPS requests to enable hardware sleeping. For instance, if the preferred location update interval is 1 minute, the application can register/unregister the request every one minute. Assuming unregistering Gps will turn GPS off, this method does not rely on the underlying framework to support

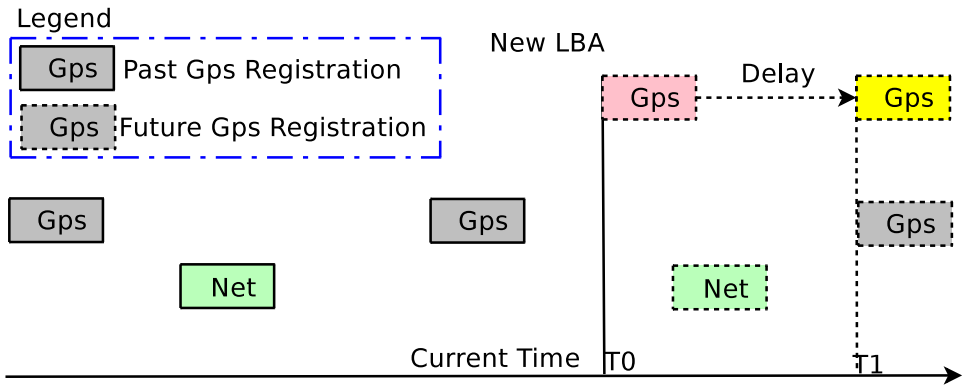


Figure 41: Sensing Piggybacking

energy conservation through GPS sleeping. The downside of this method is the increased complexity of application design. It needs more involvement from the applications by requiring the application to control when to start and stop location sensing. But such involvement also gives the user/application more control over when and how to perform location sensing. For instance, the user may require different degrees of accuracy and frequency when performing locations sensing in different scenarios. Such requirements are hard to satisfy with single-time registration and not supported by current APIs and systems. We refer to the first type of registration as *One-time* Registration, while the second type as *Multi-time* Registration. For One-time Registration, depending on the mobile systems, optimizations might be applied to save energy. Whether and how to apply the techniques depends on the GPS location management of multiple registrations. Specifically, when there are multiple sensing registrations, the underlying location manager needs to accommodate multiple registrations with different sensing requirements. For example, if there are two registrations with 2-minute and 1-minute update interval, respectively, the location manager may combine these two registrations by simply considering the finer one, i.e., every 1 minute.

In this work, we focus on Multi-time Registration, as mobile platforms such as Android have already employed mechanisms to synchronize the location sensing actions for One-time Registration scenarios. For Multi-time Registration, we propose to piggyback the otherwise wasteful sensing on other sensing invocations. Specifically, we present Sensing Piggybacking (SP) with respect to the following two scenarios which involve the joining of a new LBA. We assume the joining LBA has location sensing requirement of (G_1, T_1, D_1), where G_1 is

the granularity of sensing (e.g., fine (or Gps) and coarse (or Net)), T_1 is the minimum time interval and D_1 is the minimum distance interval for location updating. We also consider the cases where other applications are running when the LBA joins. We use (G_f, T_2, D_2) to denote the finest existing Gps registration, where T_2 and D_2 are the finest sensing intervals. Similarly, we use (G_c, T_3, D_3) to denote the finest Net registration.

- *The joining LBA has Gps request:* When a new Gps registration with (T_1, D_1) comes, the currently registered requests (G_f, T_2, D_2) are retrieved. (i) If Gps requests have been registered so far with (T_2, D_2) and if $(T_1, D_1) > (T_2, D_2)$, SP does not invoke sensing in response to the new request, but wait for the next sensing of (T_2, D_2) request. Statistically, the new registration request waits, on average, for $\frac{T_2}{2}$ time. If $(T_1, D_1) < (T_2, D_2)$, the new request is registered immediately. (ii) If only Net requests are registered, then SP immediately registers the new Gps request. Figure 41 illustrates one piggybacking scenario where both Gps and Net registrations have been maintained. The joining LBA requests Gps, and the new registration is delayed to piggyback on other Gps registrations.

- *The joining LBA has Net request:* When a Net registration with (T_1, D_1) comes, the current registered requests are checked. (i) If there are Net requests registered so far and $(T_1, D_1) > (T_3, D_3)$, SP waits for the firing of next sensing. On average, the new request waits for $\frac{T_3}{2}$ time. (ii) If only Gps requests are registered, then SP check to see whether Gps registrations satisfy its requirement. If so, SP uses the current one; otherwise, it registers a Net request.

4.4.4 Sensing Adaptation (SA)

There are different ways to save phone battery power and each of these focus on adapting a specific phone attribute. Such measures may include adjusting the screen light, sleep-time, or even the volume of ringtones. In this work, we focus on energy-saving methodologies in the context of location sensing.

The key idea of Sensing Adaptation (SA) principle is to adapt the location sensing frequency based on the current battery level. The main rationale behind such adaptation is

user's preference of longer phone-operating time over higher location accuracy. Except for running several accuracy-critical applications, users are most likely willing to trade accuracy for longer battery life. For instance, when the battery level is low and a user is running Twitter on his mobile phone and using the Gps for the location sensing, the user might be more willing to run the LBA with less-accuracy in return for longer phone use time.

SA is designed to respect the preference for longer operation time. When the battery level is low, SA is invoked and adapts the location sensing parameters to save energy. SA can be implemented in three ways: (i) changing the sensing frequency or interval; (ii) changing the sensing distance interval; and (iii) adjusting the aggressiveness of other design principles. The first two ways adapt the sensing intervals of location requests and registrations. For newly joining LBAs, this can be done by hooking into the registration process and directly changing the registration requests. For already-running LBAs, SA needs to remove existing registrations and add new registrations with adjusted parameter values. Specifically, when the battery level is low and the user wants to conserve energy, the sensing time intervals and distance intervals will be increased correspondingly based on two adaptation functions f_{time} and f_{dist}, respectively. Denoting the requested time update interval, distance interval, and current battery level by T_i, D_i, and L_b, respectively, T_i and D_i can be obtained by $(T_i, D_i) = (f_{time}(L_b), f_{dist}(L_b))$. Furthermore, users may be given the opportunity to manually input the desired adaptation degrees rather than using pre-defined ones. For this, the user can be greeted by a GUI interface which allows user input for controlling the adaptation degree.

4.4.5 Integrated Operation

So far we have separately described four design principles to improve energy efficiency. The four design principles can work together for better energy saving in various scenarios. We show the integrated operation for an examplary scenario in Figure 42. In the scenario, the user is initially moving and the battery level is high. After the user starts LBA-1 at time T_0, SS begins to work. After the second LBA starts at T_1, SP becomes operational. When the user becomes static, SR kicks in. When the battery level becomes low, SA comes into

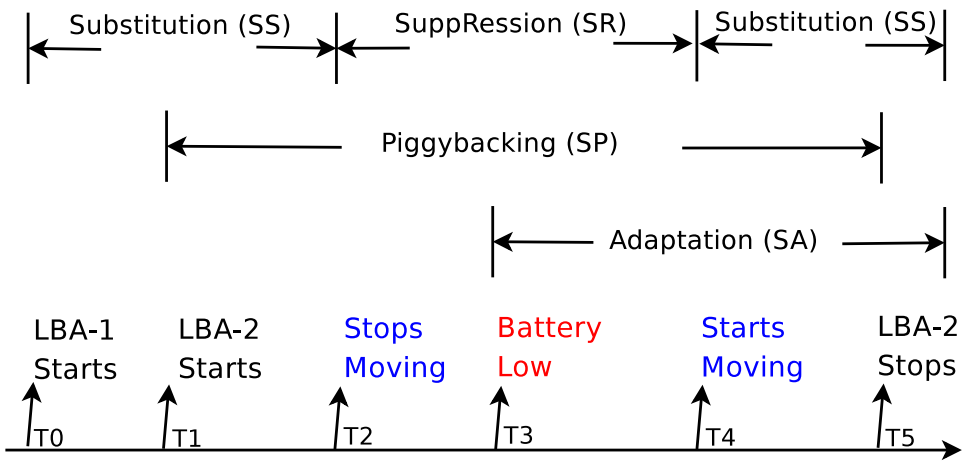


Figure 42: Integrated Operations

play. As the user starts moving again, SR stops, and SS is invoked if possible.

4.4.6 Inherent Tradeoffs

So far we have presented four design principles and their integrated operation to save energy associated with location sensing. These design principles essentially trade accuracy and timeliness of location sensing for energy saving. Along these lines, we do note that some applications might be sensitive to the location accuracy and sensing timeliness, regardless of the battery level and power consumption. Examples of such applications include healthcare and military LBAs. For these applications, all adaptation techniques have to respect application requirements. Thus, one way to safely perform the adaptation without violating the application requirement is to be application-aware and application-specific. In other words, the four design principles can be selectively adopted by application designers, when LBAs are developed. For instance, an LBA can be designed to detect the phone's mobility state and perform SR when possible. In this way, the decision about whether to apply a specific design principle and how to apply is made by the designer, and the application requirement regarding location sensing accuracy is not violated.

However, the aforementioned application-layer adoption has an associated implementation cost and is not scalable, particularly because of the vast amount of existing and future applications. Realizing this, we propose another adoption model—a middleware approach—which maintains transparency of application requirements. We will elaborate

on this functionality in Section 4.5. With the middleware approach, smartphone users are explicitly asked to decide whether to apply a design principle or not to maintain application requirement about accuracy. Practically, users may be greeted with an user-interface asking the preferred action. Users can even be given finer controls such as deciding the adaptation parameters.

The primary reason for users' involvement is to equip them with final decision-making authority. For any LBA, different users may require different levels of location accuracy. For example, given a health-care LBA, a healthy teenager may think that high location-accuracy is unnecessary, while an elder patient may think otherwise. Furthermore, even for the same application and the same user, the importance of location accuracy also vary. For instance, when a person is sick, the health-care LBA becomes more important. A more intelligent design is to remember or even predict the users' selection, thus reducing the users' overhead in such decision making. We see this enhancement as future work.

4.5 Software Architecture and System Implementation

We now present the software architecture of a system that incorporates the design principles discussed in the previous section. We explain its detailed system implementations on Android Development Phones (ADPs).

4.5.1 Architecture and Deployment Model

Even though our solution can potentially be applied to any mobile platforms that deploy location-based services, we specifically present the architecture on Android platforms for the ease of presentation and the concreteness. Such a selection is also justified by Android's open nature and increasing popularity. Note that the architecture and design principles can be applied to other platforms such as Symbian, Windows Mobile. As illustrated in Figure 43, the system is realized as a middleware solution, residing between applications and underlying Linux kernels. Specifically, Android platform includes Application Framework that packages many useful classes in Java. The solution is implemented inside the Android Application Framework by modifying existing classes as well as creating new classes. As illustrated in Figure 43, SA supplies the other three design principles with adaptation information, and

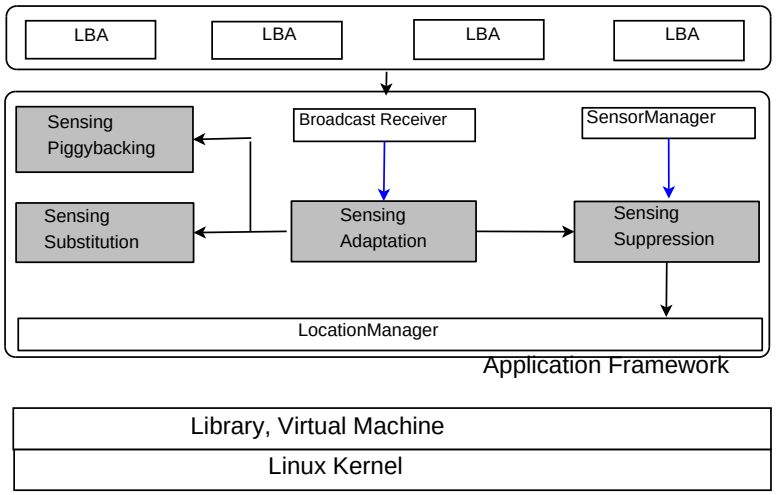


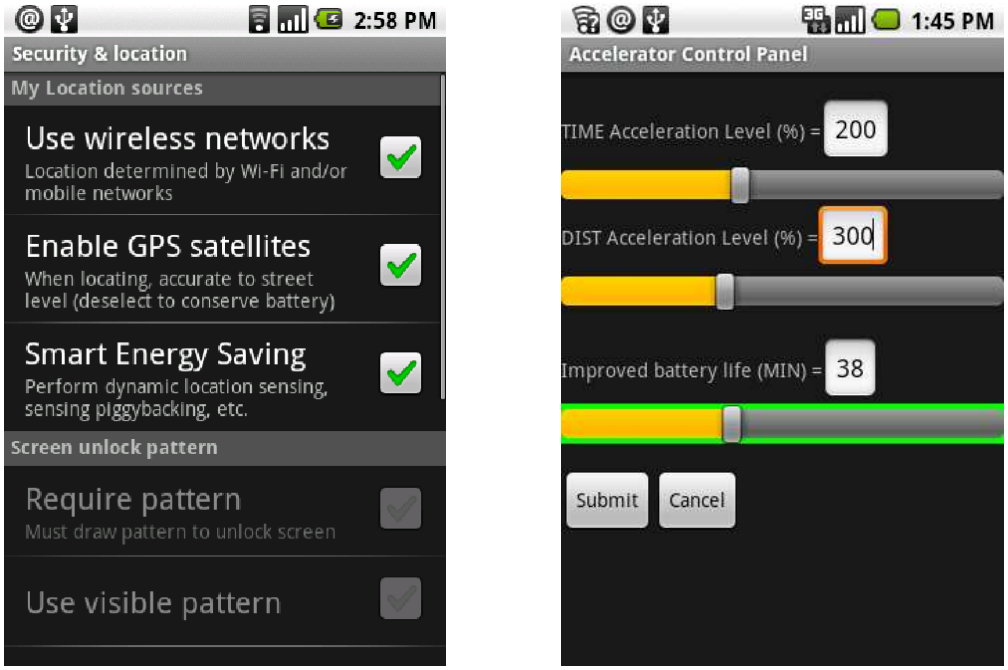
Figure 43: Software Architecture

all principles work closely with several existing components such as LocationManager and SensorManager in Android Framework.

With this deployment model, the adoption of the proposed solution on Android phones is through a new system image, which includes both new Application Framework and embedded applications. Users may choose to re-compile the source code to obtain the new system image or simply download the system image from Internet, and then update the phones with *fastboot* utility provided in Android SDK to flash the phones.

4.5.2 Implementation Overview

We prototype the proposed solution on G1 Android Developer Phone (ADP1) with OS version 1.5 Cupcake. All the four design principles are implemented in Java inside Android Framework. The prototype contains Graphic User Interface (GUI) which allows a user to enable, disable and finely configure the prototype. Figure 44(a) shows the interface for enabling/disabling the adaptive location-sensing framework. The interface is implemented inside the default "Security & location" setting menu of G1 phones. The new menu item, called "Smart Energy Saving", has been added. Figure 44(b) shows the configuration interface for the desired SA degree in time (TIME) and distance (DIST). The interface also shows the expected battery saving time with the current LBA requests and SA degrees. Briefly, the prototype first calculates the expected number of saved GPS invocations with



(a) Enable/Disable interface (b) Configuration

Figure 44: Two prototype interfaces

SA. Then, assuming a typical operation of making a phone call and its associated power level, the prototype estimates the improved battery life from the saved energy.

With current Android APIs, GPS is invoked through a major function call, requestLocationUpdates(), which takes at least four input parameters: LocationProvider (i.e., Gps or Net), reporting frequencies in term of time and distance, and an PendingIntent or LocationListener. Our prototype mainly captures this function call and embeds intelligence inside the function as well as other relevant functions. Specifically, SS may substitute another LocationProvider for the requested one, SR may freeze the further execution of the function when necessary, SP may piggyback the current call on existing registrations and freeze further execution of the function call, and SA may adjust reporting frequency based on battery level or user preference,

We illustrate the high-level operations of the four design principles, as well as the major data structure, information flows and the function calls in Figure 45. SP is hooked into the location-sensing registration function, requestLocationUpdate(). Whenever the framework detects a new location sensing registration, SP records the registrations into Registration

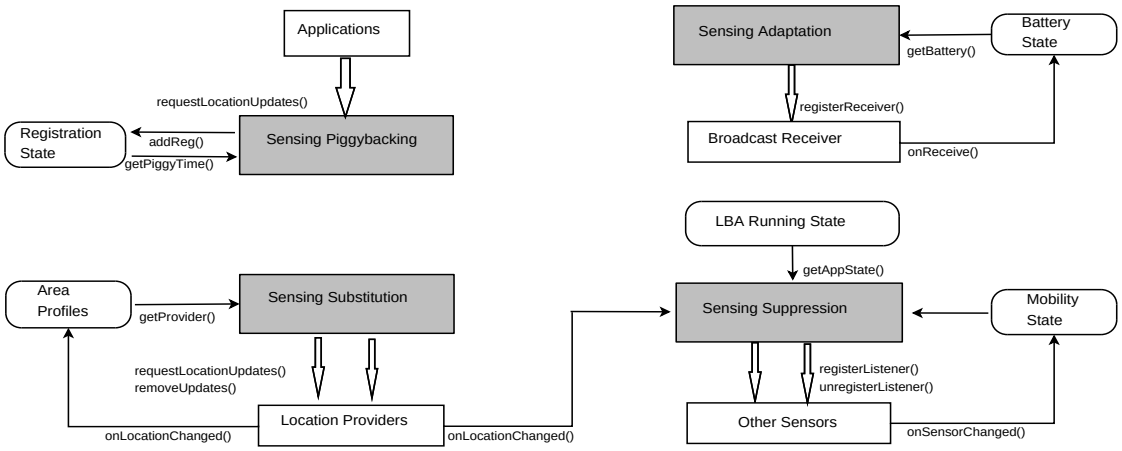


Figure 45: Prototype on G1 Android Phone

State and obtains the piggybacking time by checking this state. SA and SR are implemented in separate threads, and their invocations are triggered by battery level changes and timers. SA registers for battery change updates with Broadcast Receiver. SR periodically checks the user's mobility state for the purpose of registering or unregistering sensor readings. SS reads the state of Area Profiles, periodically determines the current M-Area and selects the most appropriate location provider.

4.5.3 Sensing Substitution (SS)

SS aims to determine the most appropriate location provider on-the-fly, irrespective of what location provider LBAs request. Specifically, when Net is available and currently Gps is being used, SS may decide to use Net to replace Gps for location sensing. The decision of whether to perform SS is controlled by the user with an pop-up dialog informing the Net accuracy and asking for actions. Similarly, when Net is being used and becomes unavailable, SS may turn to Gps. Since Gps consumes more power, Gps is requested with reduced location update frequency to maintain the same level of power consumption as the Net consumes.

In order to perform dynamic selection of location providers and accommodate the mobility of the phone, SS needs to be invoked periodically. The Handler class in Android SDK is used to implement a separate thread inside the LocationManager Class for this purpose. As shown in Figure 46(a) line 1-2, whenever the task is invoked, SS attempts to determine

the most appropriate M-Area where the phone is residing. After finding such an M-Area, SS then determines the available location provider with getProvider() call. Specifically, the prototype captures the registration of the provider, and records the registered provider, the listener, the registered time update interval and the distance interval. These information are used for new registrations (with the same interval values and the same listener). If the available provider is Net and the requested provider is Gps and if the Net can satisfy the LBA's requirement (Lines 3-4), then SS unregisters the current provider and registers the available one (Lines 5-6). If the available provider is Gps and the requested provider is Net, then SS unregisters Net and registers Gps appropriately (Lines 7-10).

Area Profiles are initialized with training data and updated by monitoring the sensed environmental characteristics when running LBAs. A separate profiler process keeps running when the user carries the phone and moves around. The process records GPS locations, network-based locations, and the time. The profiled data are stored in files, and then further extracted into M-Areas. Area Profiles are read into the memory whenever the instance of LocationManager is created. Profiled locations are organized as a list of M-Areas, each of which has the same characteristics of the two location providers. In other words, locations inside the same area has the same physical characteristics of Gps and Net (i.e., availability, accuracy, precision). The structures and operations of the M-Areas are presented in Section 4.5.7.

To reduce false negatives of area determination, the prototype uses both current location and mobility properties to decide the current M-Area. The mobility properties include current moving speed and direction. For each invocation of SS, if the current location is inside the same M-Area and if the moving direction and speed suggest that the user will be in this area for a while, then the M-Area is determined to be a candidate M-Area. If multiple candidate M-Areas exist, the most appropriate one is chosen based on a set of criteria including visiting frequency, most recent visit time and area size.

4.5.4 Sensing suppRession (SR)

SR monitors user's context with less-energy-intensive accelerometer and orientation sensors. When the user is in a static state, the prototype saves energy by suppressing the new location sensing. When LBAs are running and the location services are registered, a thread is created to monitor and identify whether the phone is in static or moving state. If the current state is static, then it removes the current location sensing registration; if the state is non-static, SR re-registers the previous sensing request, as shown in Figure 46(b) (Lines 8-12). The thread is invoked periodically (e.g., every 1 minute) and the reading for each invocation last for several seconds. The reason for doing so rather than continuous monitoring is that otherwise the continuous sensor reading and computation also consumes more energy. However, the disadvantage of periodic reading rather than persistent reading is that short-term static states might not be detected. Thus, periodic invocations work best for long-term static state.

Inside the thread, the prototype reads accelerometers and orientation sensors to detect mobility (Lines 1-7). The basic rationale is that whenever there is change of the state, these sensors will see big changed values. As the motion sensors may report updates quite frequently (e.g., 20 times per second), the user state is detected to be static only when both microscopic state and macroscopic state are static. Microscopic state is determined by finer neighboring sensor readings, while macroscopic state is determined by coarser reading changes (e.g., 2 second). We notice that both microscopic and macroscopic checking are necessary since there are scenarios where slow change (i.e., macroscopic) is happening, but such changes cannot be detected by microscopic checking. For instance, the most infrequent sensor reading rate (i.e., by supplying SENSOR_DELAY_NORMAL in the registerListener() call) on Android platform is about 10-20 times per second, as observed by our experiments. When the state change is slow, simply comparing two continuous readings is not able to detect the state change. Furthermore, to reduce the false negative (i.e., mobility being detected as being static) probability, our prototype takes one step further. If no mobility is detected, then the user state is considered to be *transiently* static, and this state has to sustain for certain period before inferring that the state is static.

4.5.5 Sensing Piggybacking (SP)

LBAs request the location sensing through a registration function call of requestLocationUpdates(), which takes several parameters including the location provider, time interval and distance interval. The essential idea of SP is to force the incoming registration request to synchronize with existing location-sensing registrations. SP predicts the next sensing registration request from currently running LBAs and asks the incoming LBA to delay the registration. SP learns and maintains the location-sensing registration history, stored in two array lists—one for Gps and the other for Net. Each element of the lists contains three values: registration time, time interval and distance interval.

SP firstly needs to determines the validity of the maintained states. Since the prediction of future registrations is based on historically maintained states, the state might be invalid as it can be outdated because the requesting LBAs might stop running or change the registration. A state is valid only when the most recent registration time recorded is no more than certain time earlier than the current time. The default threshold value for determining the validity is 200% of the time interval. In other words, if the predicted registration which is supposed to occur after T time does not come in $2T$ time, then the state is invalid, indicating either the application changed the registration pattern or the application has stopped running.

As shown in Figure 46(c), SP is hooked into the registerLocationUpdate() function in the LocationManager Class of Android Framework. When receiving the above function call, SP checks the validity of the maintained registration state (Lines 1-2). If the state is invalid, the request is passed through and is added to the registration history by the addReg() function. If the state is valid, then SP determines the piggybacking time (i.e., the delay) with getPiggyTime() function (Lines 4-16). The current prototype determines the piggybacking time in six different usage scenarios, based on the currently maintained registration-state types as well as the incoming new registration type. In the following, we will discuss each of the six scenarios below. For simplicity, we use the notation of {(Maintained states), Incoming state} to denote each scenario. We use (t, T_0, D_0) to denote the incoming request, where t is the time, T_0 is the requested update time interval, and D_0

is the requested distance interval. For the maintained states, we use (Gps, T_1, D_1) to denote the Gps state with the finest time interval being T_1 and finest distance interval being D_1. We use (Net, T_2, D_2) to denote the Net state with the finest time interval being T_2 and the finest distance interval being D_2.

- {(Gps), Gps}: The prototype checks whether the (Gps, T_1, D_1) state is valid. If so, then it compares (T_1, D_1) to (T_0, D_0). If $T_1 < T_0$ and $D_1 < D_0$, then piggybacking is enabled, and the piggybacking time is calculated.

- {(Gps), Net}: As Net typically has coarser location information than Gps, the operations are similar to the ({Gps},Gps) scenario, but the comparison is between (T_2, D_2) and (T_0, D_0).

- {(Net), Net}: Similar to {(Gps), Gps} case by replacing Gps with Net.

- {(Net), Gps}: Since Gps is typically finer than Net, the request cannot piggyback on existing Net registrations. The new registration is passed through immediately.

- {(Gps, Net), Gps}: Similar to that of {(Gps), Gps}.

- {(Gps,Net), Net}: The prototype firstly checks the Net state, which is similar to that of {(Net), Net}. If not possible to piggyback, then it checks the Gps state, which is similar to {(Gps), Net} scenario.

4.5.6 Sensing Adaptation (SA)

The operations of SA are shown in Figure 46(d). SA is invoked when Gps is used and the phone's battery level is low. When the battery level is below a user-specified threshold (e.g. 20%), SA determines the preferred adaptation degree for both time and distance intervals of Gps registrations (Lines 1-3). SA also asks a user's intention on whether to perform SA or not. If adaptation is enabled, the user can choose the preferred adaptation degrees. The prototype then functions based on the decision and values provided by the user (Lines 4-7).

SA learns the current battery level information with Android power-APIs. It registers a BroadcastReceiver to handle the Intent of ACTION_BATTERY_CHANGED. The function

used to register is registerReceiver(), which is a method of the Context class in Android SDK. Because of this, the prototype piggybacks the registration on an existing application in Android platform: SecuritySettings, which is extended from Context. Specifically, in the onCreate() method, SecuritySettings registers the BroadcastReceiver and an IntentFilter. Whenever the battery level changes, the receiver is notified and appropriate information is recorded.

Applications running on Android platforms are essentially independent in the sense that each application has a private directory and each application runs in a separate Java virtual machine. For communications between activities within a single application and between different applications, Android SDKs provide several mechanisms including shared preferences, content providers and database. Unfortunately, none of these mechanisms works neatly for the communication between application layer and framework layer. Our prototype uses files (under /proc) as the intermediate media for these two layers to communicate. Specifically, applications and frameworks both access the same files under the data directory of the system, which can be obtained by getDataDirectory() call. There are various types of data that need to be shared. For simplicity, we use a separate file for each type of data.

4.5.7 Mobility Profiling

Both SR and SS use the M-Area structure to organize the locations. Each M-Area contains three types of properties. The first type is boundary property. Each M-Area is a rectangle area bounded by a starting point, an ending point, and a width value. The points are specified with latitude and longitude coordinates. The second type is usage property. M-Areas also contain the number of visits and the last visit time (i.e., LastTime). The third type is provider property. M-Areas also maintain the sensing characteristics, such as availability and accuracy, of Gps and Net.

The construction of M-Area consists of the two steps. Initially, each M-Area is constructed as a rectangle, based on the two neighboring location readings from the mobility traces. Later, M-Area can be merged and replaced. Two M-Areas can merge into one when they have compatible boundary-related properties and same provider-related properties.

There are two types of merging scenarios: Horizontal and Vertical. Horizontal merging occurs when the starting point of one M-Area is adjacent to the ending point of the other M-Area or the starting point is inside of the other M-Area. Vertical merging occurs when the two neighboring areas have adjacent starting-points and ending points. When conditions are met, merging is performed and the properties of the new M-Area are updated. The two merging operations are illustrated in Figure 47. Specifically, the starting/ending points and the width are updated to represent the new M-Area. The LastTime is updated to the more recent LastTime of the previous two M-Areas, and the Frequency is set to be the average of the two Frequency values.

One important design issue is the size of the profiled M-Areas. Since the size impacts the efficiency of processing speed and suppression effectiveness, there is a performance tradeoff with regard to the number of M-Areas maintained. Specifically, increasing the size results in higher suppression probability. However, it also occupies more storage space and inflates the processing time. We propose to adjust this size based on the hardware capability of the smartphones. If smartphones can afford to provide more space and process the operations sufficiently fast, maintaining in general more M-Areas benefits Sensing Substitution. In addition, replacement mechanism that only maintain higher-utility M-Areas can be easily applied to alleviate the storage concern and maintain scalability. The prioritization is enforced in the following order: Frequency, LastTime, and Area size.

4.6 Performance Evaluation

We evaluated the effectiveness of our prototype. We first model the energy saving when each of the four design principles is being applied. Then, we show the effectiveness of each design principle by considering a typical scenario where each design principle works. Finally, we evaluate the integrated operations of the prototype and show its aggregated saving.

4.6.1 Analysis

We analyze energy-saving benefits coming from reduced GPS invocations. For simplicity, we assume that LBAs request r number of GPS invocations per hour by default and that the energy cost of per-GPS invocation is E_g. Similarly, we use E_n to denote the energy

cost of per-Net invocation, and use E_o to denote the energy cost of running each design principle in an hour. The energy-saving benefits are expressed in the reduced number of GPS invocations and, for a more concrete understanding, they are translated into the extended battery life when other tasks are performed. Specifically, we choose the representative task of making phone calls, and the power consumption level of the task is denoted by P_c. We use N_g to denote the number of GPS invocations reduced by each design principle in an hour, and use T_c to denote the extended operation time, when making calls.

- *Sensing Substitution* Assuming p_u percentage of GPS invocations are replaced by Net invocations:

$$N_g = rp_u \text{ , and } T_c = \frac{rp_u(E_g - E_n) - E_o}{P_c}$$

- *Sensing Suppression* Assuming p_s percentage of GPS invocations are suppressed, we have,

$$N_g = rp_s \text{ , and } T_c = \frac{rp_sE_g - E_o}{P_c}$$

- *Sensing Piggybacking* Assuming p_g percentage of otherwise-independent GPS invocations can piggyback on other invocations. We have,

$$N_g = rp_g \text{ , and } T_c = \frac{rp_gE_g - E_o}{P_c}$$

- *Sensing Adaptation* Assuming the time-interval adaptation degree is $d_t(\%)$, we have,

$$N_g = r(1 - \frac{100}{d_t}) \text{ , and} \quad (1)$$

$$T_c = \frac{r(d_t - 100)E_g - E_o}{d_tP_c} \quad (2)$$

We now show exemplary values based on experiments and assumptions mentioned above. Our measurements show that each GPS invocation costs about 9 Joules (i.e., 150 mA$\times$3.7V$\times$15 seconds). The average energy overhead of running the design principles on our smartphone prototype is negligible (i.e., a few mW), compared to GPS sensing power, so for simplicity in the following presentation we ignore this cost. Next, though the power level of making phone calls varies on different phones and conversation scenarios, we choose an averaged value of 600 mW, measured in an ADP. For an LBA requesting Gps every

half minute, we have $r = 120$. Thus, with SA and $d_t = 300$, the energy saved per hour is $E_s = 120 \times \frac{2}{3} \times 9 = 720$ (J) with Equation 1. We have $T_c = \frac{720}{0.6} = 1200$ (seconds) with Equation 2. In other words, with SA, for every hour of running an LBA, about 20 minutes of phone-call time can be saved.

4.6.2 Sensing Substitution (SS)

We evaluate SS by asking a person carrying a smartphone to walk along a route. The route is manually split into four areas with pre-defined different characteristics of Gps and Net. For ease of evaluation, we pre-set the characteristics of the four areas as follows. In Area 1, both Gps and Net are available, with Net being much less accurate than Gps. In Area 2, both Gps and Net are available, with Net having accuracy similar to Gps. In Area 3, only Gps is available. In Area 4, only Net is available. We run an LBA requesting Gps updates every 5 seconds. The substitution checking thread uses an interval of 15 seconds. We then record the events of SS and location updates in Figure 48. As shown in the figure, in Area 1, Gps is used to perform location updating. As the user moves into Area 2, since Net has accuracy similar to Gps's, Gps is replaced by Net. Then, as the user moves into Area 3, the component substitutes Gps for Net, since only Gps is available. Finally, when in Area 4, Net again replaces Gps to perform location sensing.

Figure 49 shows the recorded GPS invocation times and improved battery life in our experiments. Because SS replaces Gps with Net only when Net provides the desired location sensing accuracy, we vary the location accuracy required by LBAs from 50 meters to 300 meters. We set the Net accuracy according to the traces collected from a particular user who commutes along a walking route. The user lives and works in Bay Area of California, USA. As shown in the figures, with coarser requirements, the number of GPS invocations decrease. While 50-meter accuracy requirement does not see much improvement, 300-meter requirement effectively reduces the number of invocations by about 50%. Correspondingly, improved call-making time increases as accuracy requirements become coarser.

4.6.3 Sensing suppRession (SR)

SR is invoked only when the phone is in static state. We consider a scenario where an LBA is running and user's mobility states vary between being static and moving. The State-Checking thread is invoked every 1 minute. Figure 50 shows the various events such as thread invocations, starting and stopping of the application, and the user's mobility. As shown in the figure, the phone is initially static. After LBA starts, accelerometer is invoked. Since the phone is not moving, the State-Checking thread puts the phone into a suppression mode, after a while. Once the phone starts moving, the thread detects the mobility and takes the phone out of the suppression mode. Finally, after the application stops, the accelerometer is unregistered.

Figure 51 shows the recorded GPS usage with varying GPS intervals requested by LBAs. Note that we put the phone into static state for half of the entire period (i.e., 30 minutes in a hour). We plot the improved battery life when making calls in Figure 51(b). SR effectively suppresses about half of the GPS sensing, which improves the battery life when making calls by up to 400 seconds.

4.6.4 Sensing Piggybacking (SP)

SP can help reduce the number of GPS invocations by piggybacking GPS sensing requests from multiple LBAs. We run two LBAs concurrently but with different starting time. Both applications are requesting GPS sensing every 2 minutes. Figures 52(a) and (b) show the sensing updates received by the two applications. We see that when SP is not working, GPS is invoked totally 10 times in 10 minutes, while when SP is used, GPS is only invoked 6 times. Note that in Figure 52(b) the last two GPS invocations notify both applications about the new location updates.

Figures 53(a) and (b) show the GPS invocation times and improved battery life time during experiments. We vary the GPS requesting frequencies of LBAs from every 1 minute to every 3.5 minutes. With SP, the number of GPS invocations is reduced by half, and correspondingly, call-making time is improved by up to 910 seconds.

4.6.5 Sensing Adaptation (SA)

We evaluate SA by considering two scenarios with different battery levels. We set the adaptation degree for time interval to $d_t = 200$. The Gps location updates received by the applications are plotted in Figure 54. As shown in the figure, the LBA requests the location sensing updates every 1 minute, and with this component, the update interval is increased to every 2 minutes.

Figure 55(a) shows the GPS invocation times at low battery level. We vary the adaptation degree from 100% (i.e., without SA) to 350%. We observe that no-adapting results in about 60 times of GPS sensing, as requested by the applications. The higher adaptation degree results in the less number of GPS invocations, and specifically, with $d_t = 350$, GPS is only invoked 15 times. As shown in Figure 55(b), SA helps improve call-making time by up to 650 seconds per hour.

4.6.6 Integrated Results

We also evaluate the effectiveness of integration operations in energy saving. We run two LBAs concurrently at low battery level to enable corresponding components of SA and SP. The adaptation degree is set to be 200%. The two LBAs request GPS sensing with same frequency of every 30 seconds, but start with 15-second difference. We use the traces collected from a particular user who commutes along a route. We also vary the user states to invoke the SR. Specifically, we vary the time length of the user being static. The GPS usages are plotted in Figures 56. We see that by default GPS is invoked about 240 times per hour. By invoking all the four components, GPS invocations can be reduced to about one-fifth even when the phone is constantly moving (i.e., SR is not invoked). Even more significant reduction on the number of GPS invocations can be achieved when the phone is put in longer static state (i.e., up to 98%). Also, with our prototype, improved call-making time is more than 2,700 seconds for all considered scenarios.

Even though the above evaluation results show the savings in terms of GPS invocation times and predicted operation time, it is also necessary to show the improved battery life since operating the design components (e.g. computation) also consumes energy. We show

the improved battery life with our prototype with a scenario where two LBAs are running, each requesting GPS every 1 minute. The two LBAs start with 30-second difference. To show the effect of SA, we invoke the component for all battery levels, i.e., the battery level threshold is set to 100%. A user carries the phone and walks along the commuting route with different moving/static time. As shown in Figure 57(a), our prototype can improve the battery life from 81% to 92% after an hour.

We also use the LBA of Real Time Traffic to measure the effectiveness of our prototype. Using the same configurations as described in Section 4.3. The user carrying the phone follows the commuting route and spends half time walking and half time being static. The instantaneous battery level results are shown in Figure 57(b). We observe that our prototype can improve the battery life from 79% to 88% after an hour—up to 75% improvement.

4.6.7 Profiling results

To evaluate the location-sensing characteristic profiler in SS, we ask three users to carry phones with our prototype installed, and we continuously obtain their location information on a daily basis for 3 weeks. The users live and work in the Bay Area of California, U.S.A. We show part of a M-Area map both before and after the merging operations in Figure 58. We see that there are totally 5 M-Areas before merging, and these areas result in 3 new M-Areas after merging.

The profiling process has several pre-defined parameters for extracting and merging M-Areas. One of the parameters is the initial width of extracted M-Areas. The setting of this value particularly affects the merging operations since only adjacent M-Areas can be merged. A larger width value encourages merging and leads to smaller M-Area sets, while the accuracy of the M-Area extraction might be compromised since all the locations inside the same M-Area are supposed to have the same characteristics. As shown in Figure 59(a), setting the width to 10 meters rather than 30 meters increases the resulting M-Area set by more than 70%.

We also measure the Net accuracy with profiled data of 3 users, and we show CDF in Figure 59(b). We see that for the locations visited, more than 70% of locations have

a Net-accuracy finer than 100 meters. This suggests that for an LBA requiring location accuracy coarser than 100 meters, SS can be invoked most of time.

4.7 Related Work

Recently the use of smartphones (e.g., iPhones, Windows Mobile, Symbian and Android) becomes pervasive. These smartphones are equipped with location sensing capability to enable LBAs. But to the best of our knowledge, the systems don't employ techniques similar to our designs to improve energy efficiency of LBAs. Users are increasingly adopting a wide variety of LBAs on smartphones [9, 20, 34]. Several research efforts exist regarding the design and use of LBAs. For example, work in [68,72,88,121] presents traffic monitoring designs. BikeNet [61] describes an extensible mobile sensing system for cyclist experience mapping. StarTrack [41] extracts users' sequences of locations in the form of tracks so that other applications can take advantage of the information. Other works aim to improve the performance of positioning mechanisms such as GPS. For instance, Skyhook [31] improves the response time of positioning by combining the unique benefits of GPS, Cell Tower triangulation and WiFi Positioning.

Since typical smartphones are equipped with multiple types of sensors, applications that take advantage of these sensors are booming, and many existing works attempt to detect and extract users' states and context based on the readings from these sensors [43,49,65,80]. Many approaches are proposed to combine the information obtained from sensors including Bluetooth, Accelerometer, Audio, Camera and GPS [50, 64, 75, 86, 117].

Realizing the battery shortage problem of mobile systems, various solutions have been proposed to save energy [40, 107]. The challenges and general approaches for energy management on handheld devices are described in [115]. Turducken [110] presents a hierarchical power management architecture for mobile systems.

To address the power consumption problem of GPS sensing, some works attempt to trade accuracy of GPS for energy [45, 54, 63, 82, 91]. Work [45] proposes to use accelerometers to sense movements for saving energy, and the mechanism bears similarity with Sensing suppRession. ENloc [54] addresses the optimal location sensing problem given an energy

budget. Micro-Blog [63] proposes to balance the competing goals of accurate location coordinates and long battery life by infrequently using more accurate, but power-hungry localization services such as WiFi to offset the error introduced by less accurate, but more power-efficient localization services (e.g., GSM localization). These two works share certain features with Sensing Adaptation. In addition, work [100] selects between two data services driven by history, which bears the idea of substitution. Parallel to our work, work [82] proposes to choose the most suitable positioning mechanism based on locations as LBAs may require different positioning accuracy depending on specifical locations, and work [91] proposes to only invoke GPS when it is available and sufficiently accurate. Though sharing certain degree of similarity with the above approaches, our work differs from them in the exact usage scenarios and detailed designs. Particularly, compared to these approaches, our work provides a comprehensive energy-saving solution tailored for smartphones running multiple LBAs, and implemented as a middleware on Android smartphones.

4.8 Conclusion

In this chapter, we consider the problem of energy efficient location-sensing on smartphones. We first identify critical factors that affect energy efficiency of location-sensing with GPS through extensive experiments. These factors are static use of location sensing mechanisms, non-use of power-efficient sensors to optimize location-sensing, lack of sensing cooperation among multiple LBAs, and unawareness of battery level. Then, we present an adaptive location-sensing framework that includes the design principles of Sensing suppRession, Sensing Substitution, Sensing Piggybacking, and Sensing Adaptation to reduce the usage of GPS in various scenarios. We implement these design principles as a middleware on Android-based smartphones by modifying the Application Framework. Our evaluation results on the implementation show that our prototype can significantly reduce the GPS usage by up to 98% and improve battery life by up to 75%.

(a) Sensing Substitution (SS)
Variables
$provider$: Requested location provider
Set_{Area}: Profiled M-Areas
$Area_{prev}$: Previous M-Area
$Area_{cur}$: Current M-Area

```
1  Obtain most recently sensed location
2  Determine Area_cur based on Set_Area
3      If provider == Gps
4          If Area_cur's Net can satisfy LBA
5              Unregister the Gps
6              Register a new Net
7      Else (// provider == Net)
8          If Gps unavailable & Net available
9              Unregister the Net
10             Register a new Gps
11         End
12     End
13 End
```

(b) Sensing suppRession (SR)
Variables
$State_{cur}$: Current state (static or moving)
$State_{prev}$: Current state (static or moving)
$State_{micro}$: Micro transient state
$State_{macro}$: Macro transient state
$State_{Gps,Reg}$: Cur. requested Gps state

```
1  Obtain motion sensor readings
2  Determine State_micro and State_macro
3  If State_micro and State_macro == static
4      State_cur = static
5  Else
6      State_cur = moving
7  End
8  If State_prev and State_cur == static
9      Unregister the corresponding Gps
10 Else
11     Register a Gps based on State_Gps,Reg
12 End
13 State_prev = State_cur
```

(c) Sensing Piggybacking (SP)
Variables
$State_{Gps}$: Gps registration state
$State_{Net}$: Net registration state
$time$: Requested location sensing frequency
$dist$: Requested location sensing distance

```
1  Received requestLocationUpdate(provider, time,...)
2  Store information about provider, time, distance
3  Check validity of State_Gps and State_Net
4  If provider == Gps
5      Compare State_Gps to time and dist
6      If State_Gps allows piggybacking
7          Delays the registration to enable piggybacking
8      End
9  Else (//provider == Net)
10     Compare State_Net to time and dist
11     If State_Net allows piggybacking
12         Delays registration to enable piggybacking
13     Else
14         Compare State_Gps to time and dist
15         If State_Gps allows piggybacking
16             Delays the reg. to enable piggybacking
17         End
18     End
19 End
```

(d) Sensing Adaptation (SA)
Variables
Bat_{cur}: Current battery level
Bat_{thr}: Battery level threshold to trigger SA
f_{time}: Function to adjust time parameter
f_{dist}: Function to adjust distance parameter

```
1  If provider == Gps AND Bat_cur ¡ Bat_thr
2      time = time * f_time
3      dist = dist * f_dist
4      Obtain user preference
5      If SA is allowed
6          Unregister the current Gps
7          Register a new Gps with time and dist
8      End
9  End
```

Figure 46: Pseudo-code : (a) Sensing Substitution, (b) Sensing suppRession, (c) Sensing Piggybacking, and (d) Sensing Adaption

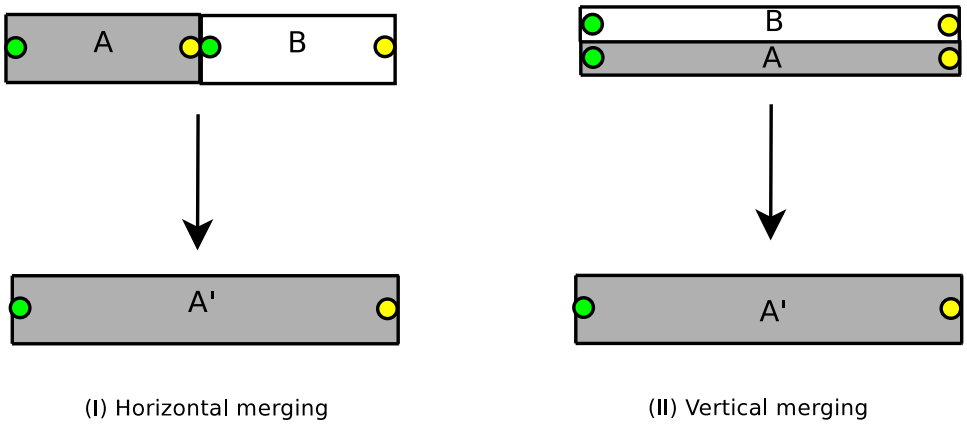


Figure 47: Merging operations

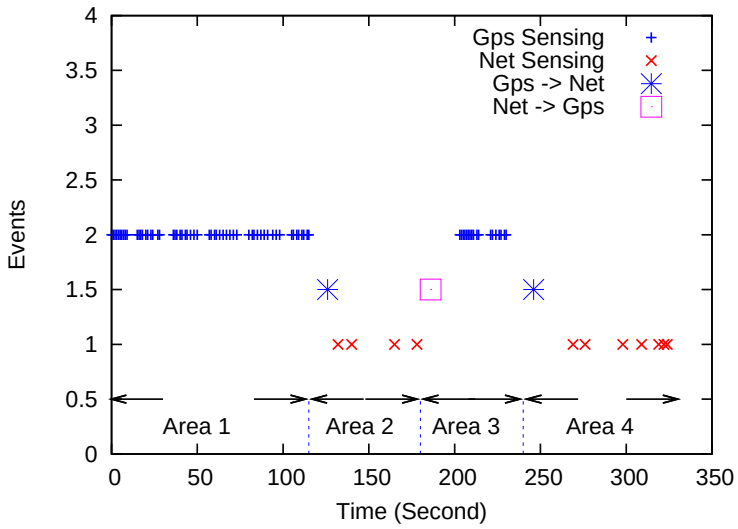
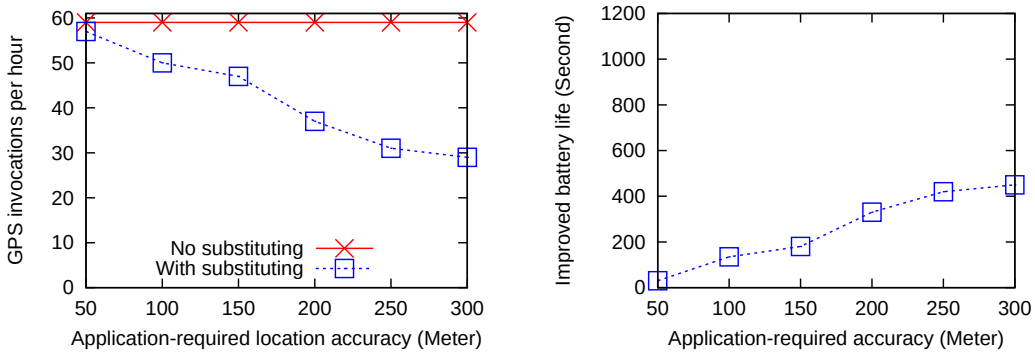


Figure 48: Sensing Substitution (Events)



(a) GPS invocation times

(b) Improved call time

Figure 49: Sensing Substitution

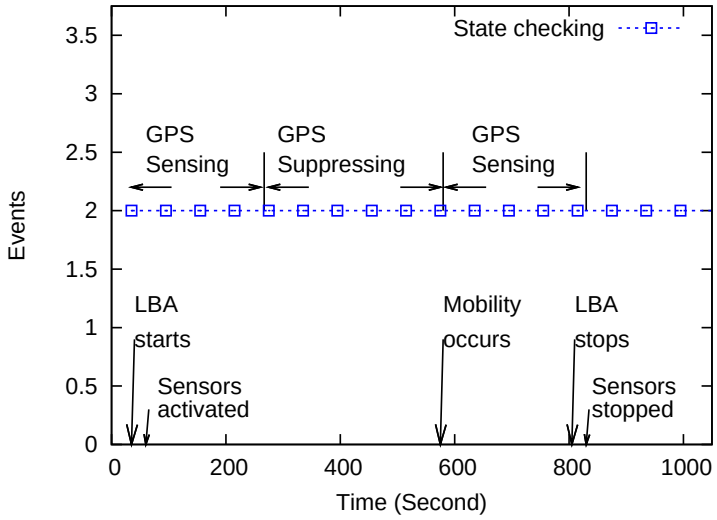
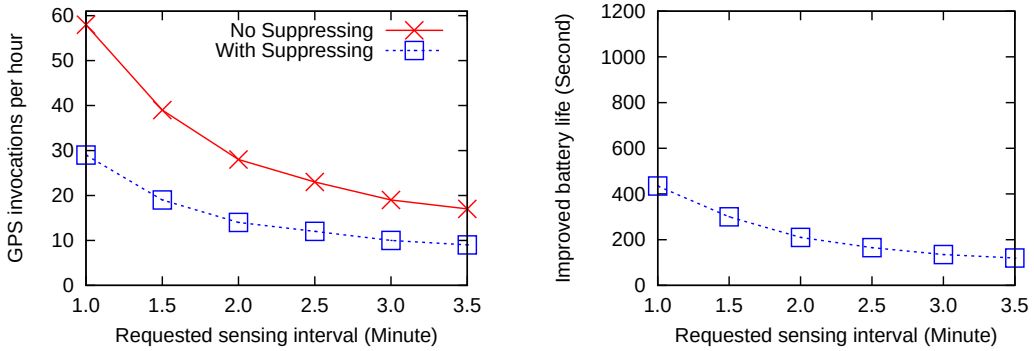


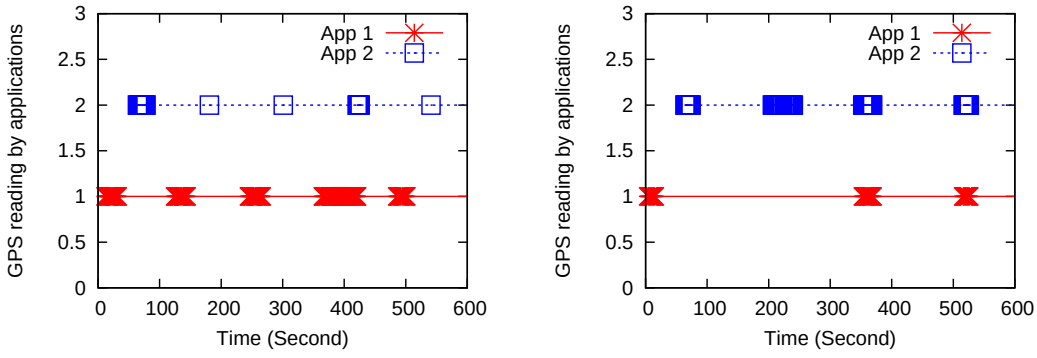
Figure 50: Sensing Suppression (Events)



(a) GPS invocation times

(b) Improved call time

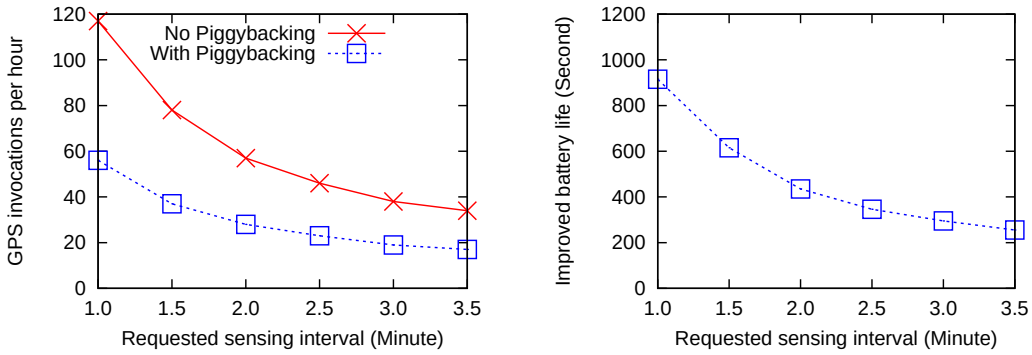
Figure 51: Sensing Suppression



(a) No piggybacking

(b) With piggybacking

Figure 52: Sensing Piggybacking (Events)



(a) GPS invocation times (b) Improved call time

Figure 53: Sensing Piggybacking

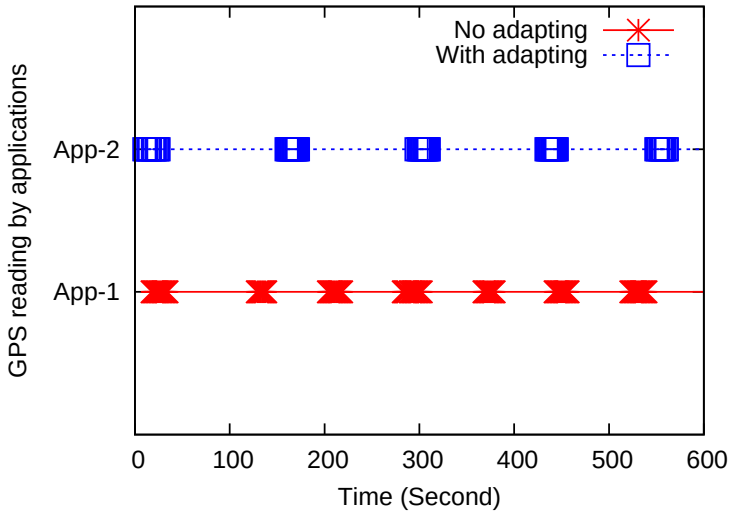
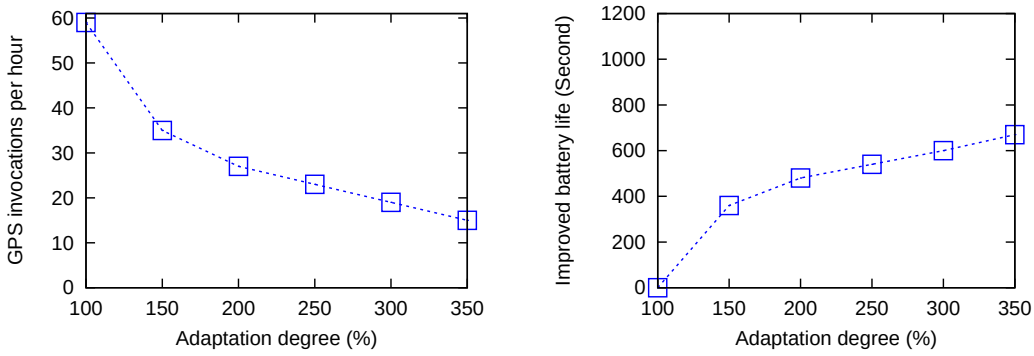
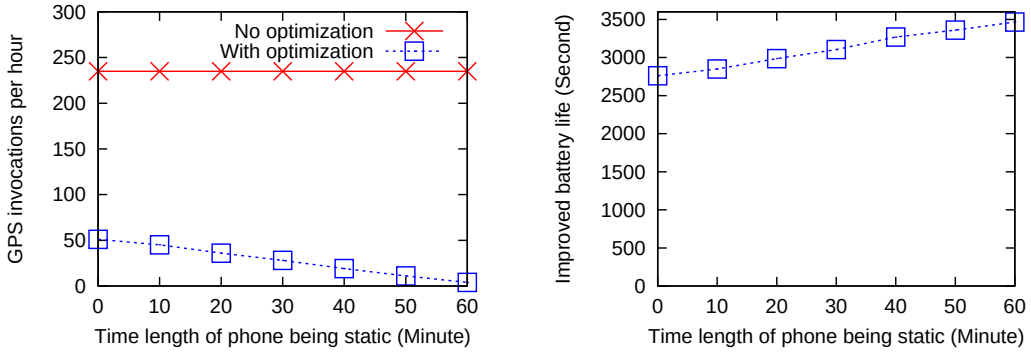


Figure 54: Sensing Adaptation (Events)



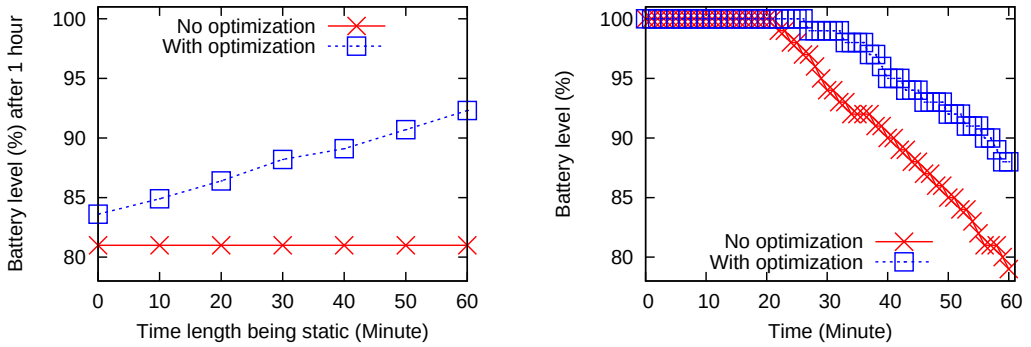
(a) GPS invocation times (b) Improved call time

Figure 55: Sensing Adaptation



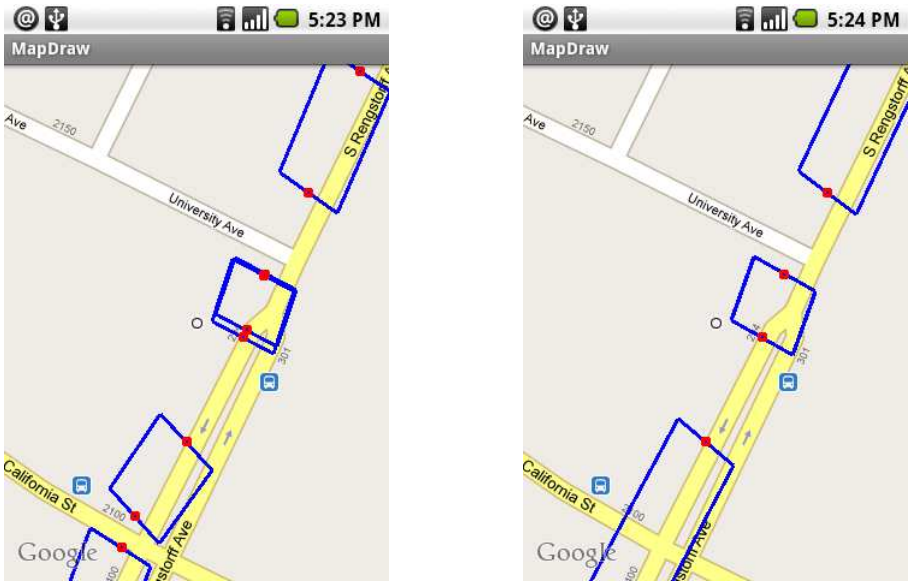
(a) GPS invocation times (b) Improved call time

Figure 56: Integrated Results



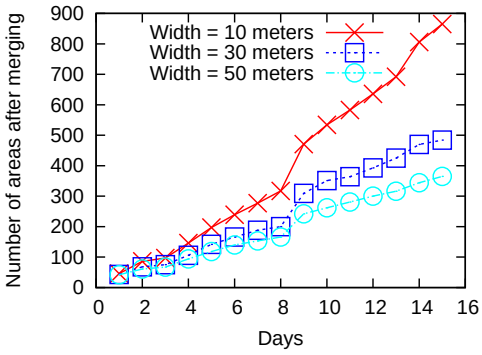
(a) Custom-built LBA (b) Real Time Traffic

Figure 57: Battery level of integrated results

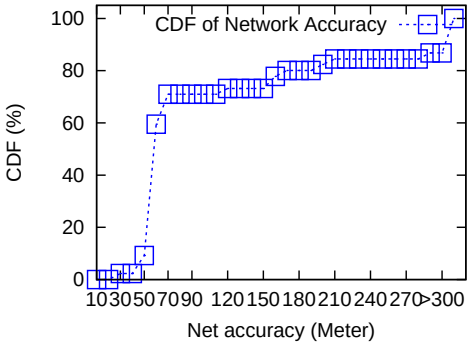


(a) Before merging (b) After merging

Figure 58: Merging operations



(a) Num. of areas



(b) Net Accuracy

Figure 59: Profiling Results

CHAPTER V

WIRELESS MEMORY: ELIMINATING COMMUNICATION REDUNDANCY IN WI-FI NETWORKS

5.1 Summary

Studies have shown the presence of considerable amounts of redundancy in Internet traffic content. Recent works are exploring possibilities for exploiting network traffic redundancy, but these works invariably focus on fixed wireline networks. Unlike wireline networks, wireless and mobile environments exhibit unique challenges and opportunities in the context of redundancy elimination.

In this chapter, we explore leveraging network traffic redundancy, but exclusively focus on wireless and mobile environments. We first analyze real Wi-Fi traces, and based on insights obtained from the analysis, we propose *Wireless Memory (WM)*, a two-ended AP-client solution to effectively exploit traffic redundancy for such environments. Our trace-driven evaluation results show that Wireless Memory can help deliver up to 93% throughput improvement for a typical Wi-Fi setup.

5.2 Introduction

Several recent studies [51,87,102,105,111] have shown the presence of considerable amounts of redundancy in Internet traffic content. Such redundancies in content can be explicitly eliminated to improve communication performance. There are various approaches [37, 38, 51, 87, 93, 94, 102, 111, 113] that have been proposed to eliminate such redundancy. Ranging from application-layer to network layer strategies, these works invariably focus on fixed wireline networks.

Similar to the above works, we too explore leveraging network traffic redundancy, but exclusively focus on wireless and mobile environments. Unlike wireline networks, wireless and mobile environments exhibit unique challenges and opportunities in the context of redundancy elimination. On one hand, the broadcast nature of wireless communication

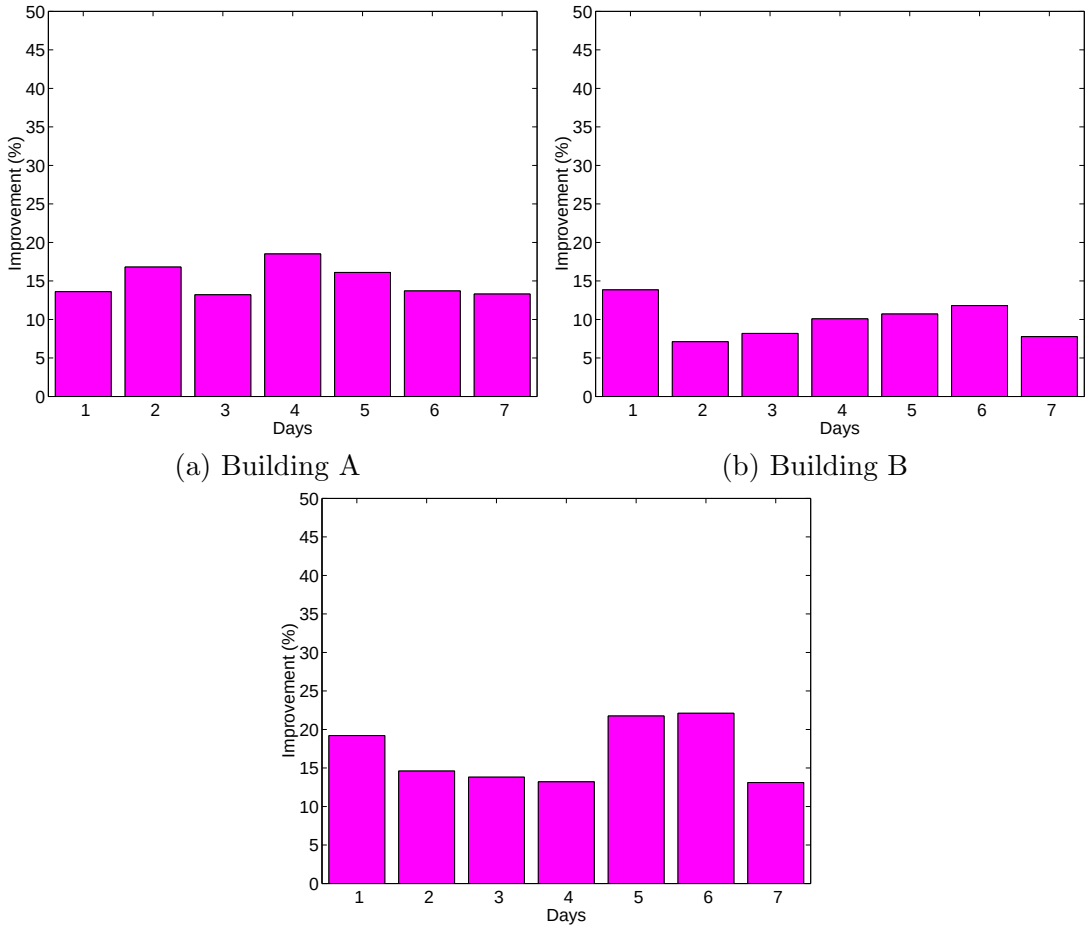
enables techniques such as packet sniffing to be performed with ease, while on the other hand, mobility and location based channel variances could impose challenges that have to be effectively addressed. Perhaps most importantly, given the typical resource constraints of wireless environments, redundancy elimination could have a profound impact on performance delivered to users.

In this chapter, we focus on one popular type of wireless networks: 802.11b/g (or Wi-Fi). We first study the traffic redundancy along multiple dimensions using traces obtained from multiple real wireless network deployments. Specifically, we consider three buildings and two Wi-Fi network deployments in a major university campus. One of the buildings is a mixed-use environment that houses several small-medium businesses. Based on the insights obtained from the analysis, we propose *Wireless Memory (WM)*, a two-ended AP-client solution to effectively exploit traffic redundancy in wireless and mobile environments. Generically, WM equips AP and clients with *memory* to enable memorization of content as it flows naturally through the wireless network, and more importantly use the memory to lower the actual cost of delivering any content to its intended destination. We evaluate WM through simulations driven by the collected Wi-Fi traces, and show that WM can improve the network throughput by up to 93% in certain scenarios.

The remaining chapter is presented as follows. In Section 5.3, we motivate the wireless memory design by presenting a set of observations and challenges. In Section 5.4, we describe the basic design and operations of wireless memory. Then we present the advanced design elements in Section 5.5. We perform trace-driven evaluation and show the results in Section 5.6. Finally we present related work and conclude the work in Section 5.7 and 5.8, respectively.

5.3 Motivation

In this section we motivate our design of Wireless Memory by analyzing collected Wi-Fi traces. The use of Wireless Memory helps only when content stored in the memory will be referenced for "future" communications. Consequently, a necessary condition for Wireless Memory to provide benefits is redundancy in traffic content. Though an extensive study of



(a) Building A

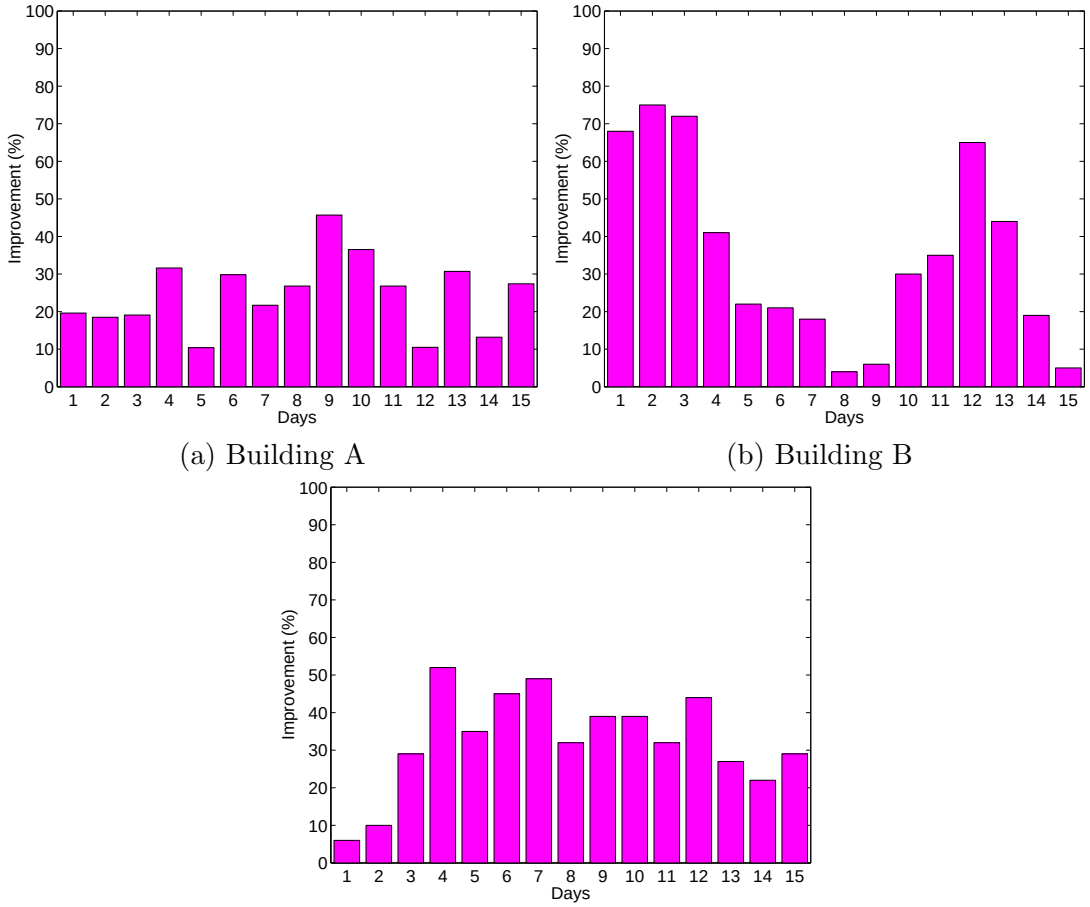
(b) Building B

(c) Building C

Figure 60: User-user dimension (Dominant user vs all other users)

the nature of redundancy in wireless traffic is a non-trivial task, we present some preliminary indicators of traffic redundancy that motivate the design of Wireless Memory. We perform studies primarily to verify that redundancy does exist for practical users. In addition, these results also shed light on our Wireless Memory design.

Data redundancy has long been observed and studied in literature, and depending on the nature of redundancy, there are two types of redundancy: *intra*-redundancy and *inter*-redundancy. It is well known that data redundancy inside a data unit (e.g., a packet) can be eliminated by applying compression mechanisms such as GZip. However, conventional compressions are unable to eliminate redundancy that exists across data units (e.g., between two packets or two html files) unless these data units are processed with the same compression session. We show that, by effectively eliminating inter-redundancy, data size can be significantly reduced, and in turn, the throughput can be improved.



(a) Building A

(b) Building B

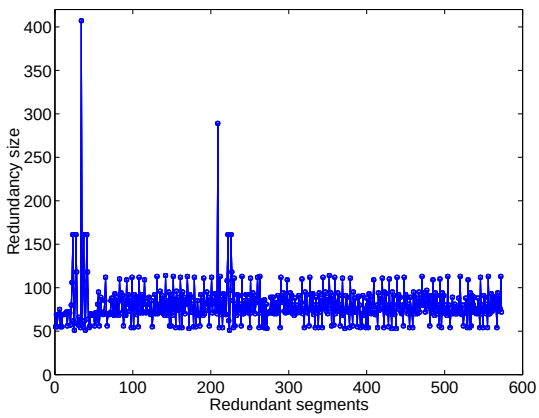
(c) Building C

Figure 61: User-time dimension (Dominant user)

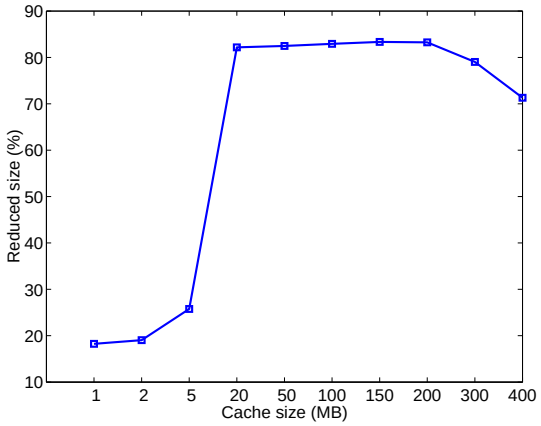
In this section we study the potential improvement on data reduction by eliminating inter-redundancy. Specifically, we compare the resultant data size of eliminating all redundancy (both intra- and inter-) to that of eliminating only intra-redundancy. For simplicity, we refer to an ideal compression that eliminates both intra- and inter- redundancy as *memory*-based compression, since the elimination of inter redundancy essentially treats the previous data as "memories". Correspondingly, we refer to the naive compressions that only remove intra-redundancy as *non-memory*-based compression.

Table 2: User-pairs (Dominant user vs. other top users)

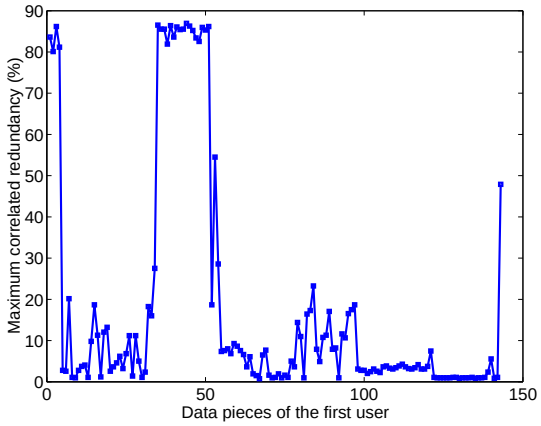
User Pair	1	2	3	4	5	6	7	8	9
Building A	12	12	14	27	7	3	19	11	17
Building B	8	13	10	7	10	14	9	27	11
Building C	49	42	33	26	17	31	29	11	8



(a) Redundancy size



(b) Dimension of memory size



(c) Clustered pattern of redundancy

Figure 62: Redundancy size (a), Memory size (b), and Clustered pattern of redundancy(c)

5.3.1 Methodology

Our study is based on real Wi-Fi traces. Specifically, we perform a 4-month wireless sniffing in 3 buildings of a major university campus. The Wi-Fi networks sniffed are two 802.11g networks, and we use four Ubuntu PCs equipped with Wi-Fi cards and MadWifi [16] in *Managed* mode. The two networks use WEP to encrypt traffic. By running MadWifi in Managed mode, the PCs are able to decode the live traffics of other wireless users in the same network. Though by associating to an AP, the sniffing desktop is able to see the decrypted traffics of other users, we explicitly perform hashing operations to store only the hash values of captured data to ensure anonymity.

To evaluate the potential improvement of eliminating inter-redundancy, we adopt the following process. First, the captured live data stream is split into packets. We consider a naive approach to redundancy-elimination by compressing each individual packets before

transmission. Note such compression is independent from application-layer compression, as the captured live traffic is the as-is traffic, and they may have been "compressed" by applications. Such a naive packet-based compression can effectively eliminate intra-redundancy (i.e., redundancy inside the packet) only. Second, an ideal approach would eliminate both intra- and inter- redundancy. So to mimic such an approach, we treat the previous packets as memory, and compress the current packet based on the memory. This method is based on the fact that most typical compression algorithms (e.g., LZW [125, 126]) process byte streams sequentially, memorize encountered byte sequences, and represent later repeated byte sequences with codes. Thus, with such a method, the packet size after eliminating both types of redundancy is estimated as the *incremental* coded size, which is the size difference of: (i) only compressing the data consisting all previous packets, and (ii) compressing the data consisting all previous packets and the current packet.

We choose a compression utility of Rzip [30] [1]. Specifically, for a particular live traffic data set, we treat the trace as a byte stream of D and split it into data pieces of d_i, where $0 \leq i \leq I$. We also use D_i to denote the set of data pieces from d_0 to d_i, so we have $D_i = \{d_0+, ..., +d_i\}$. Assuming existing compression algorithm (e.g., Rzip) can remove all intra-redundancy, the coded size of d_i is thus $Rzip(d_i)$. Similarly, the coded size of D_i is $Rzip(D_i)$. The incremental coded size of d_i is the difference of the codes of D_i and D_{i-1}. Denoting the incremental coded size is C_i, we have $C_i = Rzip(D_i) - Rzip(D_{i-1})$, and we assume C_i is the ideal coded size of d_i with memory of D_{i-1}. So compared to the non-memory-based solution which has the coded size of $Rzip(d_i)$, the effectiveness of an ideal memory-based solution is measured by $1 - \frac{C_i}{Rzip(d_i)} = 1 - \frac{Rzip(D_i) - Rzip(D_{i-1})}{Rzip(d_i)}$. The value also shows the degree of data reduction by using memory-based solution when compared to the non-memory-based solution, and the larger the value is, the higher benefit can be achieved by eliminating inter-redundancy. For all the following results, we choose $d = 1.5KB$ for the simple reason that a typical IP packet is about that size.

In the following, we will study the potential improvements of an ideal memory-based

[1] RZip is a huge-scale compression software designed to find and encode duplicated data over very long distances (e.g., 900 MB) in the input file.

approach in multiple dimensions including user-user, user-time, memory size, redundancy distribution, and application/protoco/data-types. We correspondingly show representative results which will be used to motivate our design.

5.3.2 User-user Redundancy

User-user dimension studies the potential benefits of eliminating the redundancy between users. Briefly, considering a Wi-Fi network, for a particular user U_k, other users' data can be used as the base for compressing U_k's data. We analyze the potential improvement of compressing each user's trace by eliminating the user-user redundancy with other users. Specifically, given a data piece of d_i which contains multiple users' traffic, considering a user U_k we denote his data as $d_{i,k}$ and other users' data as $d'_{i,k}$, so we have $d'_{i,k} = d_i - d_{i,k}$. Similarly we denote the cumulative data that eliminate U_k's data as $D'_{i,k} = D_i - D_{i,k}$. For the particular user U_k and his data piece of $d_{i,k}$, the compressed size with non-memory-based compression is $Rzip(d_{i,k})$, and we can get the incremental coded size of $d_(i,k)$ as $C_{i,k} = Rzip(D'_{i-1,k} + d_{i,k}) - Rzip(D'_{i-1,k})$.

We show the results in Figures 60 for all 3 buildings. For each building, we choose 1-week of traces and study the dominant user (i.e., having the largest portion of traffic) by eliminating user-user redundancy between himself and all other users. We observe that *there are substantial improvements by exploiting user-user redundancy*, and the improvement ranges between 7% to 22%.

We further study the redundancy between individual users for the same day. For each of the 3 data sets, we choose the dominant user and study the redundancy between himself and the other top 9 users. The results are shown in Table 2. We see that *some user-pairs have more user-user redundancy than other user-pairs*. More studies into this dimension suggest that the results relevant to user-pairs are caused by the web access patterns of these users. Briefly, users with higher user-user redundancy tend to visit the same set of web sites.

5.3.3 User-time redundancy

We also study the temporal user-time redundancy for individual users. We consider the top user in data sets, and choose a representative 15-day period. Starting from the second day,

Table 3: Applications and protocols

Protocol	TCP	SSL	HTTP	UDP	DNS	DHCP
Improvement	19%	3%	20%	44%	47%	81%

for each day we treat the previous day' data as the base data (i.e., memory), and compress the particular day's data by eliminating the inter-redundancy between the base data and the particular day's data. The results are shown in Figure 61, we observe that *there are considerable inter-redundancy across time for individual users*, and in certain days it can be 75%.

We further study the nature of user-time redundancy by recording the byte-lengths of redundancy units that are larger than 50 bytes, and show the redundancy sizes Figure 62(a). We observe that *most of the redundancy is at sub-packet level*. Thus, these redundancy cannot be removed by application-level caching.

5.3.4 Memory size

Eliminating inter-redundancy requires memorization of repeated byte sequences, and the size of this memory affects the compression. Since every piece of redundancy requires a code to represent it, and the code size depends on the memory size (i.e., the number of memory entries), there is a performance tradeoff between the memory size and the compression size. Specifically, though memorizing more data helps eliminate more redundancy, the code size also inflates.

To study the impact of memory size, we examine the compressed size of incoming data pieces. Some representative results are shown in Figures 62(b). We fill the memory with history trace data of fixed sizes for a Wi-Fi data set. We observe that: (i) *Increasing memory size in general brings benefit in the forms of reduced coded size.* For the particular data set we studied, the coded size can be 85% smaller than naive compression. (ii) *Increasing memory size indefinitely will incur more cost, which negates the benefit.* Specifically, the performance gets worse for the particular data set after 200 MB of memory.

5.3.5 Distribution of redundancy

We also study the distribution of inter-redundancy by examining the locations of redundant byte-sequences, and we find that the *inter-redundancy shows clustered pattern*. One

representative result is shown in Figure 62(c). In the figure, we consider the data streams of two users (i.e., U_A and U_B) and split both stream into data pieces of 15 KB. For each data piece of U_A, we identify the maximum correlated data piece of U_B (i.e., the data piece showing maximum inter-redundancy with the studied U_A's data piece). We then output the amount of inter-redundancy between these two data pieces. As shown in the figure, for many data pieces of U_A, U_B has a data piece that has very high inter-redundancy with it - a pattern we refer to as "clustered" pattern.

5.3.6 Application/protocols/data types

We then study the redundancy degrees of separate applications. We consider a specific Wi-Fi data set and choose the top six protocols (applications). We show the results in Table 3 in the order of traffic percentages. These protocols are: (i) TCP (excluding HTTP and SSL), (ii) SSL, (iii) HTTP, (iv) UDP, (v) DNS, (vi) DHCP.

We observe that *different protocols/applications have different degree of improvement when eliminating inter-redundancy.* Specifically, (i) SSL sees very little improvement, this is caused by the fact that all application data are encrypted; (ii) TCP and HTTP see some amount of improvement, and the degrees depend on the redundancy patterns of upper-layer applications and data types; and (iii) UDP, DNS and DHCP see significant improvement, and most redundancies are identical byte sequences corresponding to protocol segments, such as "www.google.com" in the case of DNS and "Subnet Mask: 255.255.240.0" in the case of DHCP.

We then consider different data types (i.e., Binary/Plain text) for the same application/protocol and observe that data types also matter. For instance, we observe that *most*

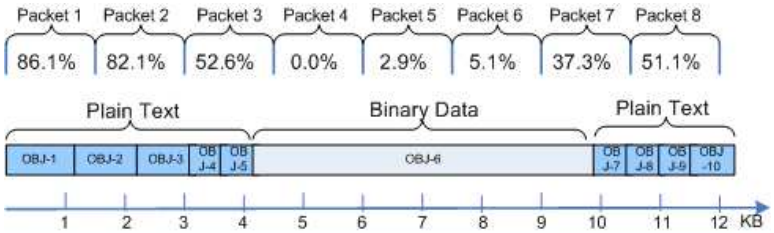


Figure 63: Dimension of data type

of inter-redundancy comes from plain text contents, while binary data such as gziped contents see little redundancy. To better understand this, we consider a representative part of HTTP trace for a particular web user, and show the improvements for 10 neighboring objects in Figure 63. These objects are either plain-text HTTP responses or binary objects. We observe that when the contents are plain texts, there are significant improvement on the reduced data size. For binary contents, little improvement is observed.

5.3.7 Summary

Traffic redundancy occur in multiple dimensions, and users' actually transmitted data sizes can be reduced by eliminating redundancy. Reduced data size will result in improved network performance including increased throughput and lower response time. This is particularly true for wireless networks, since the wireless media is often shared by multiple users, and data transmission is subject to various collision scenarios where smaller packet sizes are preferred.

5.4 Wireless Memory

We now present the solution of Wireless Memory (WM), including the concept, basic elements, and advanced elements.

5.4.1 Concept

Though many elements of WM can be applied to any wireless data networks, the basic network model we consider is the popular Wi-Fi networks with APs and mobile clients. In the following we will use Wi-Fi to describe WM. The overall benefits of using WM are better network delivery performance in terms of higher throughput, lower response time, and higher network utilization levels, through the exploitation of redundancy that naturally exists in wireless traffic.

With the basic network model, WM works between AP and clients. The simplest fashion in which the wireless memory works is as follows. Assume the AP S has to deliver certain data d to a client C at time T_1. That data is memorized by both S and C. Later at time T_2, S sends another information which contains d, S can retrieve d from the C's memory

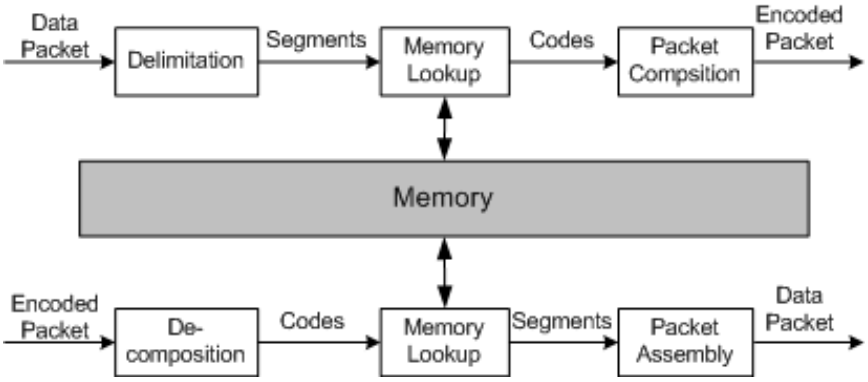


Figure 64: Basic elements of WM

by sending d's reference to C. The retrieval command sent from S to C is generally much smaller than the raw data. For the data that are not available in memory, S sends them directly.

5.4.2 Basic design elements

The basic elements of WM are illustrated in Figure 64. WM maintains memory space on both AP and clients. When AP communicates with multiple clients, it maintains separate memory for each of the clients. For any AP-client pair, their memories are synchronized in the sense that they contain identical data. The synchronization is achieved implicitly as both AP and the client see identical data being transmitted and received. In addition, identical memory operations such as data referencing and replacement will be performed.

WM works at packet-level and has two types of operations: *Memory Referencing* and *Memory De-referencing* to encode and decode the data packets, respectively. Their pseudo-codes are shown in Figure 65. Specifically, (i) Memory Referencing sequentially invokes three components of Delimitation, Memory Lookup and Packet Composition. When WM receives a data packet, it firstly delimitates (i.e., splits) the data payload into a sequence of data segments (i.e., data pieces). For each segment, it performs memory lookup to determine whether the segment is in memory or not. If present, the segment will be replaced with a code, and the code can be simply the index of the corresponding memory entry. If not present, then the segment is left as-is. After all segments are processed, a new packet will be composed containing both raw segments and codes. (ii) Memory De-referencing contains the *complementary* components of Packet De-composition, Memory Lookup and Packet

Assembly. When a coded packet is received, WM firstly separates the raw segments and codes. For each code, it performs memory lookup to recover the corresponding segment. When all segments are recovered, it assembles them to form the original data packet.

There are some other details regarding the aforementioned operations. (i) The Delimitation component is based on Rabin-based delimiters [99], which has been shown to have many advantages over fixed delimiters when used to identify redundancy [102][2]. (ii) Packet Composition results in a coded packet which consists of two regions. The first region is the data region, which contains all segments that cannot be found in memory. The second region is the code region, which consists of a list of <*code, offset*> entries. Each entry represents a redundant segment with the code and the starting offset in the data region. (iii) When Packet De-composition receives a coded packet, it can insert the recovered segment back to the data region to recover the original packet. (iv) WM works in a per-packet fashion and does not perform packet partition or packet-aggregation. Given a packet, if the coded packet is larger than or equal to the original one, WM will send the original packet. (v) When a WM-enabled client firstly associates to a WM-enabled AP, they exchange certain WM-related information to initialize and synchronize WM operations. These information include the memory space size, delimitation parameters, replacement algorithm. (vi) Though memories maintained on both sides are designed to enforce synchronization, when errors do occur (e.g., a reference being unable to decode), WM will report error back to the sender and the sender will retransmit the original packet.

5.4.3 Advanced design elements

The advanced design of WM enhances the aforementioned basic elements by six advanced components. For clarification, from now on, we will use WM to refer to the advanced design of WM, while refer to the basic elements of WM as Memory Referencing and De-referencing (MRD). The software architecture of WM is shown in Figure 66. WM is designed to be application transparent, meaning no application needs to be changed, and WM can improve the performance of all applications. We assume a design residing at layer-2.5 between the

[2]A more detailed description can be found in Appendix-5.8.

```
Variables
  P: Current packet
  Set_P: Segment set of P
  Set_C: Code set of P

  Received a packet P:
  If P is an outgoing packet
     Delimitate P into segment set of Set_P
     For each segment S in Set_P
        Do memory-lookup
        If Cache-hit
           Reference the segment with code
           Update the corresponding memory entry
        Else (// Cache miss)
           Create and enqueue the memory entry
     End
  Else (// Incoming packet )
     Extract code set Set_C
     For each code C in Set_C
        Do memory-lookup
        If found in memory
           De-referencing the code
           Update the memory entry
        Else (// Cache miss)
           Report error back to sender
  End
```

Figure 65: Pseudo code for Basic WM Elements

Network layer and Link layer.

Though we defer presenting the detailed design of the six advanced components to Section 5.5, briefly: (i) Memory Filter (MF) filters out data that have low redundancy to conserve memory space. (ii) Memory Fidelity Enhancer (MFE) allows clients to eliminate user-user redundancy by overhearing other clients' traffic. (iii) Memory Localizer (ML) further reduces the code size by using localized references. (iv) Memory Replacer (MR) helps only maintain most-useful memory entries. (v) Memory Sizer (MS) determines the optimal memory size to maximize compression effectiveness; and (vi) Memory Advertiser (MA) pro-actively tells the sender that the local host has certain memory entries so that the sender can remove more redundancy.

Components of MF, MR and MS have same operations on both sides. The identical operations are essential to synchronized memory states. For instance, both sides need to filter out the same type of traffic to ensure memory synchronization. The operations of other components, which include MRD, MFE, ML and MA, *complement* each other on both sides. These complementary are necessary to ensure correct coding/decoding. For instance, on the sender side, redundant data are identified and represented with codes, while on the receiver side, they are decoded to recover the original data.

We now briefly describe the overall operations on both sides. For simplicity, we only consider the downlink traffic, with AP sending and clients receiving. The processing of the other direction of traffic only differs slightly.

5.4.3.1 AP

When AP receives a packet destined for a client from the data source, it first determines whether the packet should be filtered out or not. If so, it is let through without further processing. Otherwise, the packet will be delimited into segments with MRD and replaced with codes when possible. If succeeding segments result in cache-hit and they belong to the same cluster, then a smaller *localized* version of code will be used to replace the original code. The size of the memory is determined by MS and set to a optimal value so that the resultant packet size can be minimized. If the current memory space is full, then appropriate replacement will be done by MR to maintain highest-utility memory entries.

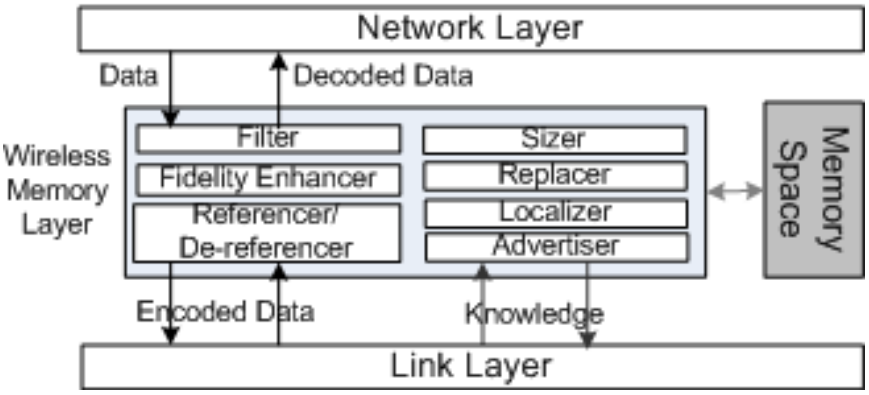


Figure 66: Software Architecture

MFE will determine whether other clients are able to overhear this packet. If another client can overhear the packet, the corresponding segments will be put into the client's memory. Finally, MA processes advertisement information from a client and enqueue corresponding memory entries from other clients' memories into the particular client.

5.4.3.2 Clients

When clients receives a encoded packet from AP, MRD extracts the codes and decode them back to original segments. This process also involves the ML to obtain the original codes. The operations of MS and MR are the same as AP. MFE requires clients to actively overhear other clients' traffic whenever possible, and the overheard data will be put into its memory. If the client notices that the segments being received form clusters, then MA will pro-actively notify AP its other memory entries that are in the same cluster but are not known by AP.

5.4.4 Memory structure

The memory structure maintained by clients and AP are shown in Figure 67. AP maintains separate memory for each associated client. Memory is split into two parts: (i) Main Memory and (ii) Shadow Memory. Main Memory of clients and AP are synchronized and contain the identical information. At any time, the encoding and decoding is performed based on Main Memory of the AP and the client. Shadow Memory consists of two types of traffics: local traffic and fidelity-based traffic. Local traffic refers to the previously passed traffic between AP and the client, but they are not used for coding. Fidelity-based traffic refers to the traffic that belong to other AP-client pairs. AP and clients may have different views of fidelity-based traffic, and part of these data can be moved to main memory after synchronizing between both sides. We will elaborate on this when describing MFE.

Memory space consists of a set of memory entries, and each memory entry contains important information including data segment, code, MD5 hash, and other useful information such as frequency. A field of Synchronization-bit (Syn-Bit) is used to indicate the validity of fidelity-based entries, and a "true" value indicates that both AP and the client has that entry and will be moved to Main Memory.

5.5 Design of advanced elements

We now present the design highlights and the operations of each of the advanced components.

5.5.1 Memory Filter (MF)

The purpose of Memory Filter (MF) is to reduce the processing overhead of other modules by removing some of the received data that result in low compression effectiveness. Encrypted data, for instance, are expected to show very little inter-redundancy and intra-redundancy, which is also confirmed in Section 5.3. Thus, these data should be filtered out. We also observe in Section 5.3 that binary data typically contain little intra-redundancy just like encrypted data, but they may exhibit inter-redundancy when the same data are transmitted more than once. Thus, binary data are removed only when the memory space is full. In addition, in Section 5.3 we observe that users exhibit different degrees of user-user redundancy and some of the users are more *correlated* than others. Thereby when space is a concern, only traffic belonging to highly correlated users will be further processed.

The pseudo-code of MF is shown in Figure 68(a). MF can be configured to filter out all three types of data: (i) Data encrypted by upper-layer protocols such as SSL, and they can be identified by protocol port numbers (e.g., TCP 443) or application-specific intelligence. (ii) Binary data such as compressed files and pictures. The identification of binary data is through the byte values, as text bytes are almost always between 0 and 127. (iii) Data belong to less correlated users. The correlation levels can be calculated based on history data, assuming users exhibit consistent access patterns.

5.5.2 Memory Fidelity Enhancer (MFE)

Since user-user redundancy exists, clients can exploit such redundancy to reduce the sizes of their own data. Memory Fidelity Enhancer (MFE) is designed for this purpose. To learn about other users' traffic, a client should explicitly sniff network traffic. Such sniffing is a trivial task in non-encrypted networks, as all data are transmitted openly. For encrypted

traffics such as WEP-based, a client can still easily decode other users' raw data. [3]

After a client learns other users' traffic, in order to encode his traffic with AP, AP needs to know what the particular client has overheard. Though a straightforward solution is to ask clients to acknowledge the packets overheard, the additional traffic associated with such acknowledgement makes the approach prohibitive. Instead, MFE allows the AP to "intelligently" estimate what clients overhear which clients' downloading traffic. The pseudo-code is shown in Figure 68(b). Specifically, each client C_i has an associated data rate R_i which is determined solely by the channel situation. Since clients can always overhear and decode traffics that are sent with lower rates, a client C_j can learn another client C_k's traffic provided that $R_j \geq R_k$. Since AP knows each client's rate, it can estimate the overhearing capability of each client at any time. The advantage of such technique is that it eliminates the necessity of explicitly verifying the overhearing results. In scenarios where this technique incorrectly estimates the results and results in non-decodable data, appropriate error handling techniques can be applied to retransmit original data.

5.5.3 Memory Sizer (MS)

The memory size impacts the coding efficiency in two conflicting ways. On one hand, a larger memory leads to more redundancy being exploited. On the other hand, the code size also inflates with increased memory, as the code size is expected to be in the order of $log(M)$, where M is the number of memory entries. To strike the performance tradeoff on the code size and the removed amount of redundancy, the size of main memory needs to be optimized so that the resulting data size can be minimized. For this, appropriate memory resizing operations are required.

Memory Sizer (MS) can deterministically compute the ideal memory size based on the past redundancy exploitation statistics. The operations are displayed in Figure 68(c). Specifically, assuming during past period, each repeated segment M_i $(1 \leq i \leq I)$ has associated properties of usage probability p_i and data size s_i. Also assuming the number of memory elements is M, so the code size is log_M. The optimal M value can be determined by

[3] For more advanced encryptions such as WPA, further efforts are required and we see this as future work.

maximizing the following effectiveness expression which counts the aggregate reduced data amount: $\sum_{i=1}^{I} p_i(s_i - log_M)$. If the algorithm keeps track of p_i and s_i, then it can easily determines the optimal M value. To reduce the performance load of memory sizing, MS is only triggered periodically or when the effectiveness of WM is below certain threshold.

5.5.4 Memory Localizer (ML)

Since the nature of WM is to use short codes to replace long segments, it is desirable to use shortest codes. In its default form, each code is a *global* value in the sense that each code can be decoded independently. To further reduce code size, Memory Localizer (ML) is designed to use *localized* shorter code rather than global longer value.

ML works as shown in Figure 68(d), and segments are firstly split into clusters based on receiving time. For redundant segments (i.e., repeated ones), since they may belong to more than one clusters, they might be tagged using an array of cluster IDs. Whenever continuous cache-hit segments are found to belong to the same cluster, they are referenced using an offset code value rather than a complete code. Specifically, the offset can be any value that allows the receiving end to decode the segment uniquely. Since both ends maintain identical Main Memory, the offset value can be much small when compared to the code size. For instance, the offset can be the last few digits of the segment's hash value.

5.5.5 Memory Replacer (MR)

Both storage space and performance (e.g., memory lookup) may put constraints on memory size. With limited memory, it is desirable to only maintain most-useful memory entries to maximize redundancy exploitation. Thus, Memory Replacer (MR) (as shown in Figure 68(e)) is used to replace less-useful memory entries by more-useful ones. Though a straightforward approach to replacement is to simply adopt popular algorithms such as Least Frequently Used (LRU), the special properties of WM requires a more advanced design. Specifically, MR works by keeping track of each memory entry's utility. For this, each memory entry is associated with an utility value that represents its "usefulness" in terms of how much data can be saved by using it. The utility is calculated as the saved bytes over the past period of time, i.e., $p_i(s_i - log_M)$.

5.5.6 Memory Advertiser (MA)

As clients may be mobile and associate to multiple APs at different time, when a client associates with an AP, either side may have accumulated certain memory but does know whether the other side has it also or not. Such memory cannot be utilized unless its existence can be verified on both ends. As WM creates synchronized memory states only based on past in-network data, to make these out-of-network memory useful, one side needs to explicitly inform (i.e., advertise) the other side what memory it has. Though a naive solution is to simply transmit these memory to the other end, the performance overhead is forbidding. More importantly, blindly transmitting memory does not guarantee benefit, as the advertised memory may not be used at all!

Sketched in Figure 68(f), Memory Advertiser (MA) can selectively advertise the memory in an on-demand fashion. First, observations made in Section 5.3 show that data redundancy exhibits clustered pattern. MA is only triggered when such clustered patterns are present. Second, it further only advertises relevant memory as they are more likely to be used. Third, MA advertises a short representation of each memory entry rather than the raw data. When memory advertisement is needed, MA can identify the appropriate relevant "knowledge" and propagate the knowledge to the other side. The basic idea is to monitor the cache-hit segments and cluster pattern. If there are a sequence of cache-hit segments, and they all come from the same cluster, then other segments of the same cluster will be extracted and sent to the other side.

For simplicity, the advertising amount is limited to one packet size, and each advertised segment is represented using a MD5 hash. The receiving side will extract these hashes and locate the corresponding segments. If the segments are found and they are not in Main Memory, they will be enqueued into the corresponding Main Memory. There is also error-processing operations. Briefly, if the receiving side detects error due to reasons including failure of locating a segment, then it will notify the sending side.

5.6 Performance Evaluation

We evaluate WM with trace-driven simulations, and the traces are described in Section 5.3. The direct result of applying WM is the reduced data size, which in many network environments translates to higher throughput. Thus, the performance metrics we consider are the resultant data size and the network throughput with a typical network setup. The simulation software we use is NS2 [114], with which an 802.11 Wi-Fi network is configured. We integrate the seven components of WM inside NS2 so that the input traces can be processed by WM before being transmitted by NS2.

We compare WM to a baseline scenario where data packets are sent as-is. For WM, we evaluate each of the seven components separately, as well as the integrated solution. Except the baseline scenario, we assume all clients are WM-enabled, unless otherwise stated. To ensure consistency of various evaluations, we always use the following network setup. The 802.11 network consists of a AP and 8 wireless clients with random placement. Each of the clients sets up a single TCP connection with another fixed host behind AP. The RTT of the wired network is 60ms and bandwidth is 100Mbps. Some other important parameters about the 802.11 setup are Congestion-Window between 15 and 1023, Slot-Time 20us, SIFS 10us, Preamble-Length 72 bytes, Data-Rate 11Mbps, and no RTC/CTS.

With our collected traces, we evaluate WM along the following dimensions. First, we choose the same data sets as used in Section 5.3.2 (i.e., three buildings) and study the aggregate network throughput. Second, we study the impact of redundancy level on both coded packet size and aggregate throughput. Since our data set is very diversified in terms of varying redundancy degrees, we choose three typical users who have comparatively low, medium and high redundancy, respectively. We also study the improvement achieved by each individual WM component. Third, we evaluate the impact of the adoption curve. Briefly, though ideally WM should be deployed on all clients for maximum performance, the progressive adoption of WM brings varying degree of benefits.

5.6.1 Aggregate network throughput

We use the same data sets as described in Section 5.3.2. For each data set, we choose the top 8 users based on traffic volumes and use their traffics as the simulation inputs. The results are shown in Figure 69. We observe that the aggregate throughput for the baseline scenario is about 6.24 Mbps. For all data sets and different days, the improved throughput vary between 7.25 Mbps and 12.03 Mbps, or an improvement between 16% and 93%. The average improvement is about 40%. Note that the throughput results are the *effective* throughput as experienced by applications rather than raw throughput. Since WM can significantly reduce packet size by eliminating traffic redundancy, the effective throughput can be larger than the physical bandwidth limit of 11 Mbps.

5.6.2 Impact of redundancy level

We now examine the impact of redundancy level on both coded packet size and aggregate throughput. We choose three representative users that exhibit different levels of redundancy and use their respective data as the inputs of all clients. The redundancy level is estimated based on the averaged packet sizes when coded by MRD. Specifically, the three users have about 10%, 35% and 60% redundancy, respectively.

5.6.2.1 Low redundancy

Figures 70 show the low-redundancy results. From Figure 70(a), we observe that though MRD can effectively reduce the data size by 11%, complete solution of WM can achieve 17% of reduction. Figure 70(b) shows the corresponding aggregate throughput. We see that WM improves the default throughput of 6.24 Mbps by more than 11%, or 6.90 Mbps.

5.6.2.2 Medium redundancy

Figures 71 show the medium-redundancy results. We observe that WM can achieve more than 40% of data size reduction. The throughput improvement, as seen from Figure 70(b), is about 32% (i.e., 8.22 Mbps vs. 6.24 Mbps).

5.6.2.3 High redundancy

Figures 72 show the high-redundancy results. We see that WM can reduce the packet size by about 62%, and the throughput improvement is about 57% (i.e., 9.79 Mbps vs. 6.24 Mbps).

5.6.3 Adoption curve

WM allows incremental deployment by mobile users. We now evaluate the impact of deployment status using aggregate throughput as the performance metric. We use the above network but with varying degree of redundancy levels, and show the aggregated throughput in Figure 73. We see that as more users adopt WM, the aggregate throughput almost linearly increases. Specifically, when the redundancy is as high as 60%, the complete adoption of WM can achieve about 54% of throughput improvement.

5.7 Related Works

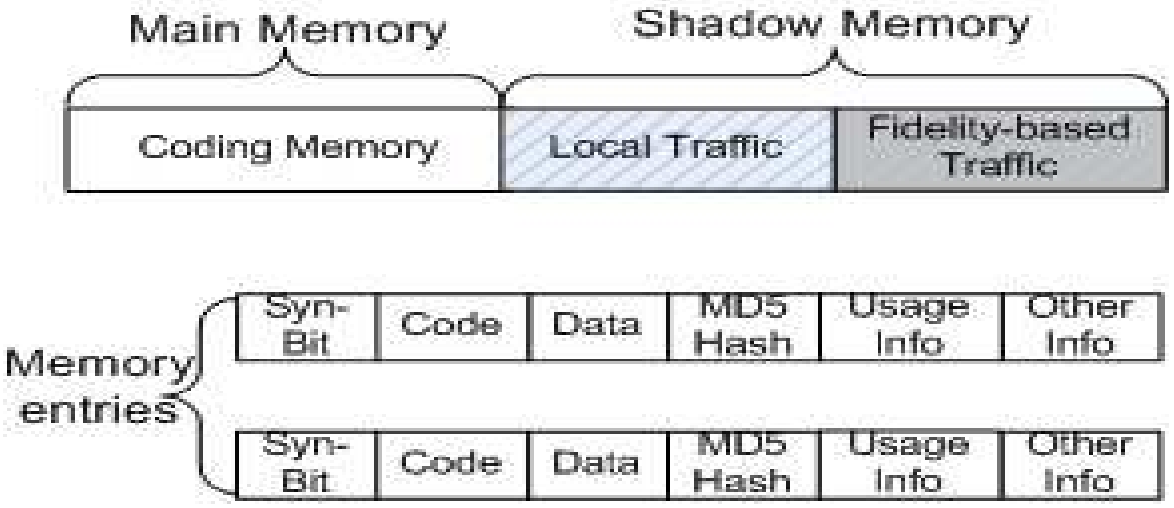
Primarily motivated by the temporal dimension of traffic redundancy on Internet, several approaches are proposed to exploit such redundancy and reduce users' response time. Squirrel [74] provides a decentralized, peer-to-peer web cache by enabling web browsers on desktop machines to share their local caches and form an efficient and scalable web cache. A churn-resistant peer-to-peer web caching system [83] is designed to resist churn attacks. [104] develops a novel caching algorithm for P2P traffic. These caching are performed on file-level, which significantly limit their effectiveness.

Various approaches are also proposed to eliminate traffic redundancy at finer granularity than packet-level. [51] proposes an efficient selection algorithm for selecting similar objects as references. A value-based web caching [102] is motivated by the facts that web files may be changed gradually and aliased, and proposes to split files into blocks. Also, a protocol-independent technique [111] proposes a mechanism to detect repetitive traffic on a communication link and provides a protocol-independent idea to eliminate the repetitive segments. [105] uses digests for packets to directly suppress redundant transfers in networks by using a proxy on either end of a low bandwidth connection. Work [37] proposes to

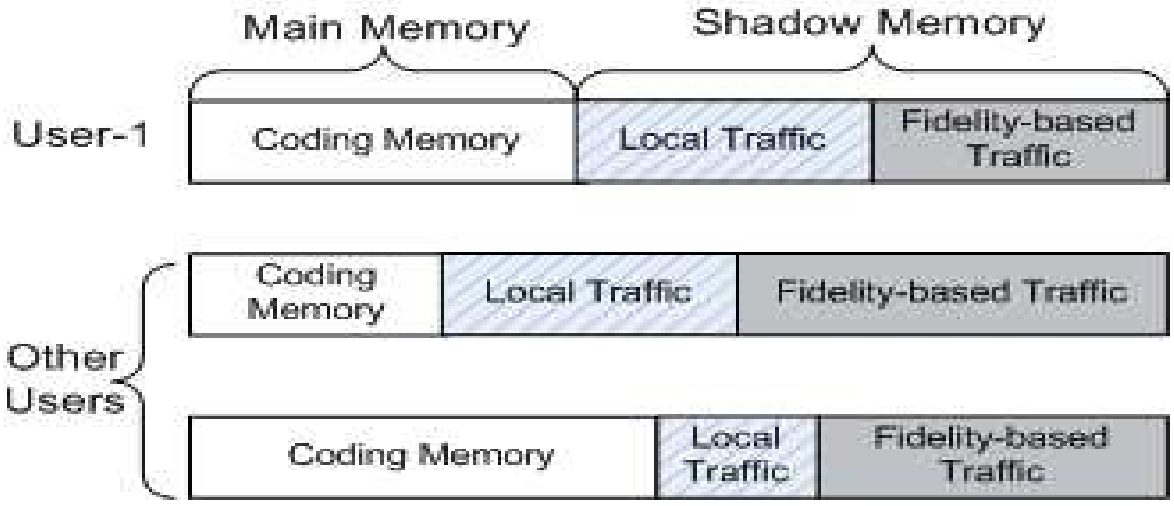
deploy packet-level memories on Internet routers and change routing protocols to explicitly remove redundancy, and Work [38] further presents redundancy-elimination design as a network-wide service.

5.8 Conclusion

In this work, we study traffic redundancy in wireless networks. Motivated by several unique observations obtained from trace analysis, we propose a solution suite called Wireless Memory which can help deliver better performance by eliminating redundancy.



(a) Client memory



(b) AP memory

Figure 67: Memory structure for clients and AP

(a) Memory Filter (MF)

Received a packet P
If P is encrypted
Let it through (filtered out)
End
If Memory is full
If P contains binary data
Let it through
End
If P belongs to non-correlated clients
Let it through
End
End

(b) Memory Fidelity Enhancer (MFE)

AP

Received a packet P from client C_i:
Obtain the rates of clients R_i
For every associated clients C_j
If $R_j \geq R_i$
Put P into j's shadow memory
End
Put P into i's shadow memory

Clients

Received a packet P from client AP:
Put P into shadow memory

(c) Memory Sizer (MS)

I: Number of main memory entries
D_{thrd}: Number of packets triggering resizing
D_{curr}: Number of packets so far

Received a packet P:
D_{curr} ++
If $D_{curr} \geq D_{thd}$
Sort main memory based on $p_i(s_i - log_I)$
Find $\hat{I}$ to maximize $\sum_{i=1}^{I} p_i(s_i - log_I)$
$D_{curr} = 0$
End

(d) Memory Localizer (ML)

$Seg_{curr,prev}$: Current, previous segment
ID_{clus}: Current cluster ID
$Array_{ID_i}$: Array of cluster IDs of a memory entry i

Delimited segs put into time-based clusters
Do memory lookup for Seg_{curr}
If Seg_{curr} results in a cache-miss
Tag Seg_{curr} with ID_{clus}
Else (// cache-hit)
Add ID_{clus} into $Array_{ID}$ of Seg_{curr}
If $Array_{IDs}$ of Seg_{prev} and Seg_{curr} share elements
Reference Seg_{curr} using decoding offset
End

(e) Memory Replacer (MR)

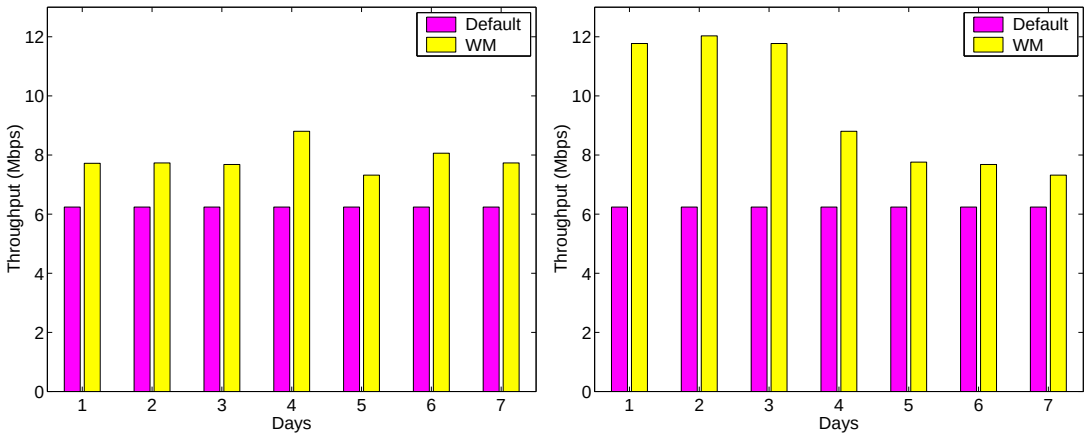
Do memory lookup for Seg_{curr}
If cache-miss
If Main Memory not full
Enqueue it to Main Memory
Else
Enqueue it to Shadow Memory
Replace least useful entry with Seg_{curr}
End
Else (//cache-hit)
Update the status of hit memory entry
End

(f) Memory Advertiser (MA)

Num_{hit}: Current number of continuous cache-hits
Thd_{hit}: Threshold value of triggering
$Array_{ID_i}$: Array of cluster IDs of a entry i
$Cluster_{max}, Freq_{max}$: Most common cluster and its frequency
Set_{adv}: The segment set for advertising

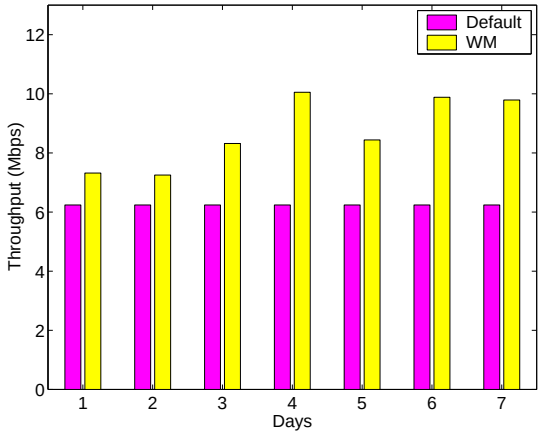
Do memory lookup for Seg_{curr}
If cache-hit
Num_{hit} ++
If $Num_{hit} \geq Thd_{hit}$ AND $Freq_{max} \geq \frac{Num_{hit}}{2}$ for $Array_{IDs}$
Get the optimal Set_{segm} of $Cluster_{max}$
Compose an advertizement packet based on hashes of Set_{adv}
Send the adv. packet to receiver
$Num_{hit} = 0$
End
Else (//cache miss)
Stop advertising memory; $Num_{hit} = 0$
End

Figure 68: Pseudo code for Advanced Design Elements: MF (a), MFE (b), MS (c), ML (d), MR (e) and MA (f)



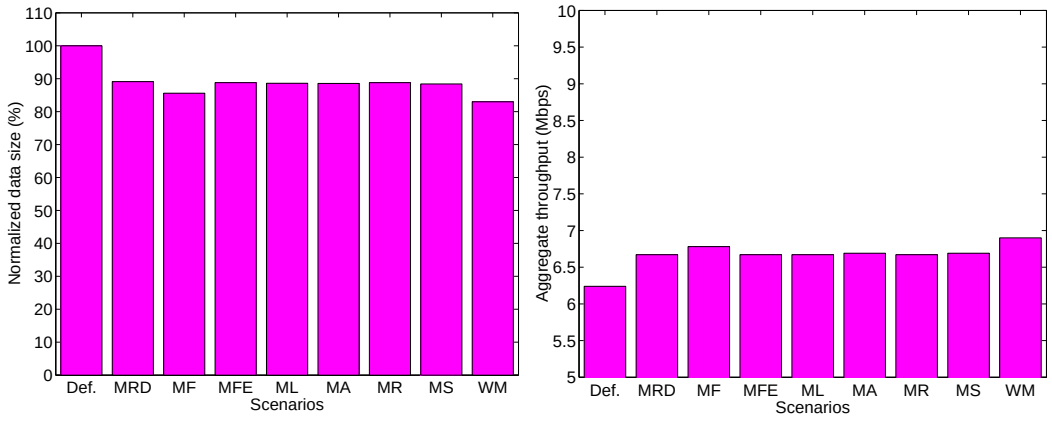
(a) Building A

(b) Building B



(c) Building C

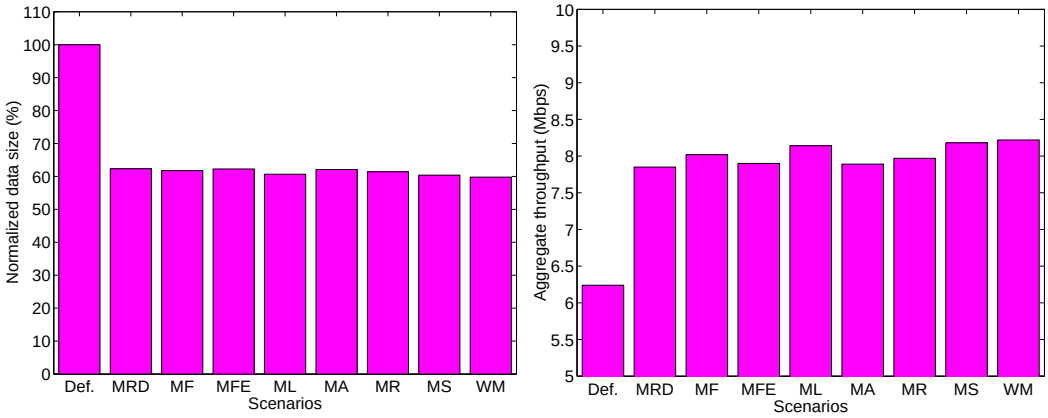
Figure 69: Aggregate network throughput based on three data sets



(a) Coded size

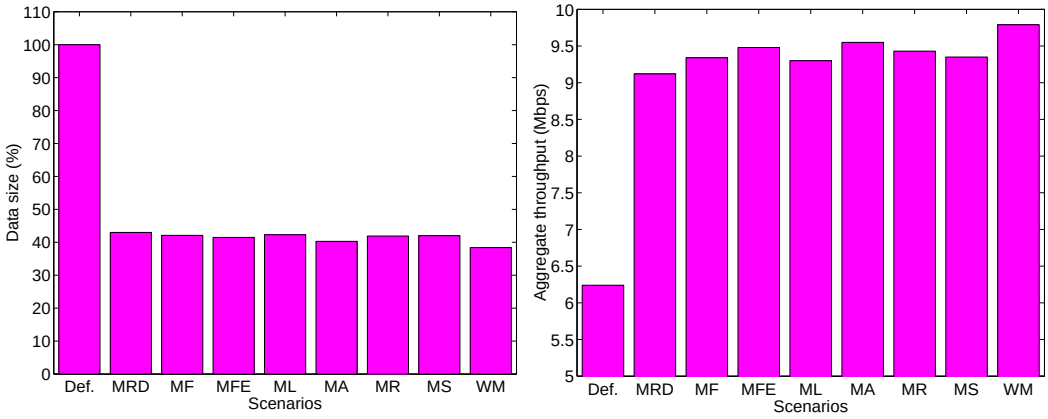
(b) Throughput

Figure 70: Low redundancy



(a) Coded size (b) Throughput

Figure 71: Medium redundancy



(a) Coded size (b) Throughput

Figure 72: High redundancy

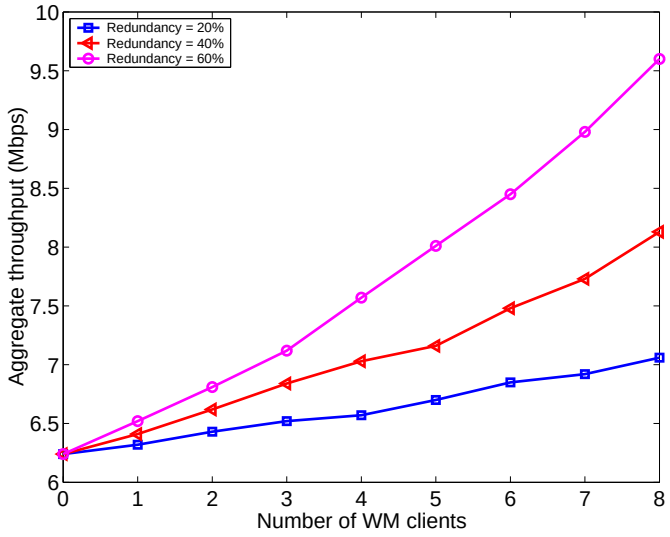


Figure 73: Adoption curve

Rabin-based Delimitation Based on Rabin random polynomials, such an approach in essence randomizes the occurrence pattern of delimiters by using a sequence of byte contents rather than a single byte value. Specifically, given a data packet of N bytes, byte B_i's Rabin value R_i is computed using $R_i = B_i K^r + B_{i+1} K^{r-1} + \ldots + B_{i+k}$ (Mod M), where r is a small integer, M a modulus, and K a prime (*e.g.* 11). The number of data segments is controlled by the delimiter probability p. By comparing R_i to pM, if $R_i < pM$, byte B_i is a delimiter, otherwise it is not. Segments then are extracted after the delimiters are identified. Specifically, if B_i is a delimiter, then it serves as the starting byte of a new segment. The segment contains all the bytes from B_i to the byte right before the next delimiter. The first and last blocks of a data packet contain the bytes left by other segments.

CHAPTER VI

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this thesis, we study the problem of application acceleration for wireless and mobile environments. We explore the problem along five dimensions: advanced network protocol design, overcoming application behavior, eliminating traffic redundancy, network provisioning and quality of service (QoS). Though tremendous amount of researches have been conducted along the three dimensions of advanced network protocol design, network provisioning and quality of service (QoS), the application performance improvement delivered by these researches are fundamentally constrained by certain properties and challenges relevant to the wireless and mobile environments.

In this work we focus on the two dimensions of overcoming application behavior and eliminating traffic redundancy. First, we consider specific types of applications including client-server applications, peer-to-peer applications and location based applications on smartphones. We identify various application behaviors that negatively impact application performance and propose design principles to deal with them. We also conduct system research by building protocols for these applications.

Second, we improve application performances by accelerating content delivery by eliminating traffic redundancy in wireless networks. The proposed solution is referred to as wireless memory, which can maintain memory on both ends of communication paths. By eliminating redundancy, traffic sizes can be reduced and the application throughput can be improved.

In summary, my thesis involves the experimental analysis of the new dimensions of application behavior and traffic redundancy for performance optimization in wireless data networks, and design of application-acceleration and wireless memory solutions guided by a strong systems-focus to exploit those dimensions using generalized principles derived from the analysis.

6.2 Future Work

Based on our conducted research, we feel that the topic of application acceleration deserves further study due to its importance and complexity. In the following, we identify four potential directions of future work. The first direction is along the dimension of overcoming application behavior. Though we have identified a set of application behaviors that affect performance, given the vast number of existing and future application types, we do not claim the completeness of our research along this dimension. We believe other application behaviors can be potentially identified and extracted from various types of applications.

The second direction is along the the dimension of eliminating traffic redundancy. We admit that our work is limited from certain perspectives and believe more researches can be done with regard to this dimension. For instance, though our approach can eliminate traffic redundancy for un-encrypted traffic, it fails to eliminate the redundancy in encrypted traffic. More advanced designs can be potentially proposed to overcome this inefficiency, possibly by being aware of the encryption mechanisms on trusted network entities.

The third direction is cross-dimension research. In this thesis, we characterize a set of dimensions for application acceleration in Chapter 1 and present the thesis work along two of the dimensions. Though each of the researches is conducted along a single dimension, we believe future researches can benefit from being cross-dimension. In other words, researches could span more than one dimensions for higher effectiveness. For instance, the dimensions of advanced network protocol design and eliminating traffic redundancy can be potentially integrated by designing network protocols that are redundancy-free. As an example, a redundancy-free transport protocol may expose interfaces to applications allowing applications to explicitly remove traffic redundancy before the actual transmission.

Finally, the fourth direction is to conduct more comprehensive system-related research in the context of application acceleration. For instance, each of the dimensions we identified in Chapter 1 could give rise to various system-related challenges when accelerating large-scale applications in real environments, and these challenges can only be addressed by considering the specific environmental characteristics.

REFERENCES

[1] "Android market," http://www.android.com/market, (03/2010).

[2] "Arch linux," http://www.archlinux.org/download, (04/2010).

[3] "Bittorrent," http://www.bittorrent.org/, (03/2010).

[4] "CIFS: A common internet file system." http://www.microsoft.com/ mind/1196/cifs.asp, (03/2010).

[5] "Comscore media metrix top 50 online property ranking." http://www.comscore.com/press/release.asp?press=547, (03/2010).

[6] "Converged access wan optimization." http://www.convergedaccess.com/, (03/2010).

[7] "edonkey2000," http://en.wikipedia.org/wiki/EDonkey2000, (04/2010).

[8] "Enhanced ctorrent, a lightweight c++ implementation," http://www.rahul.net/dholmes/ctorrent/, (04/2010).

[9] "Facebook," http://www.facebook.com/, (04/2010).

[10] "Fasttrack," http://en.wikipedia.org/wiki/FastTrack, (04/2010).

[11] "Fedora distribution," http://fedoraproject.org/en/get-fedora/, (05/2010).

[12] "Foursquare," http://www.foursquare.com/, (05/2010).

[13] "Gnutella protocol specification, version 0.4." http://www.clip2.com/GnutellaProtocol04.pdf, (05/2010).

[14] "Juniper networks." http://www.juniper.net/, (03/2010).

[15] "Linux magzine." http://www.linux-magazine.com/issue/15/, (03/2010).

[16] "Madwifi project," http://madwifi-project.org/, (03/2010).

[17] "Mandriva distribution," http://www.mandriva.com/en/download, (05/2010).

[18] "Melodeo's mobile phone p2p to launch," http://www.ringtonia.com/, (05/2010).

[19] "Minimo, a small, simple, powerful, innovative web browser for mobile devices." http://www.mozilla.org/projects/minimo/, (05/2010).

[20] "Myspace," http://www.myspace.com/, (05/2010).

[21] "Netfilter project," http://www.netfilter.org/, (03/2010).

[22] "A new type of radio," http://www.roadcasting.org/, (05/2010).

[23] "Open handset alliance," http://www.openhandsetalliance.com/, (05/2010).

[24] "Opentable," http://www.opentable.com/, (04/2010).

[25] "Peer-to-peer in 2005," http://www.cachelogic.com/home/pages/research/p2p2005.php, (03/2010).

[26] "Pocket internet explorer." http://www.microsoft.com/windowsmobile/, (01/2010).

[27] "Real time traffic," http://monthorin.net/tiki-index.php, (01/2010).

[28] "Rfc 3135: Performance enhancing proxies intended to mitigate link-related degradations." http://www.ietf.org/rfc/rfc3135.txt, (01/2010).

[29] "Riverbed technology." http://www.riverbed.com/, (03/2010).

[30] "Rzip," http://rzip.samba.org/, (03/2010).

[31] "Skyhook," http://www.skyhook.com/, (01/2010).

[32] "Torrent server for the fedora project," http://torrent.fedoraproject.org/, (05/2010).

[33] "Twidroid," http://www.twidroid.com/, (04/2010).

[34] "Twitter," http://www.twitter.com/, (04/2010).

[35] "Winpkfilter," http://www.ntkernel.com/, (01/2010).

[36] "IEEE 802.11 working group: the standards for wireless LANs," (www.ieee802.org), August 2004.

[37] Anand, A., Gupta, A., Akella, A., Seshan, S., and Shenker, S., "Packet caches on routers: the implications of universal redundant traffic elimination," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 219–230, 2008.

[38] Anand, A., Sekar, V., and Akella, A., "Smartre: an architecture for coordinated network-wide redundancy elimination," in *Proceedings of ACM SIGCOMM '09*, (Barcelona, Spain), 2009.

[39] Anand, M., Nightingale, E. B., and Flinn, J., "Self-tuning wireless network power management," in *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, (New York, NY, USA), 2003.

[40] Anand, M., Nightingale, E. B., and Flinn, J., "Self-tuning wireless network power management," in *Proceedings of ACM MobiCom '03*, (San Diego, CA, USA), 2003.

[41] Ananthanarayanan, G., Haridasan, M., Mohomed, I., Terry, D., and Thekkath, C. A., "Startrack: a framework for enabling track-based applications," in *Proceedings of ACM MobiSys '09*, (Kraków, Poland).

[42] Armstrong, T., Trescases, O., Amza, C., and de Lara, E., "Efficient and transparent dynamic content updates for mobile clients," in *MobiSys '06: Proceedings of the 4th international conference on Mobile systems, applications and services*, (New York, NY, USA), pp. 56–68, 2006.

[43] Azizyan, M. and Choudhury, R. R., "Surroundsense: mobile phone localization using ambient sound and light," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 13, no. 1, pp. 69–72, 2009.

[44] Balakrishnan, H. and Katz, R., "Explicit Loss Notification and Wireless Web Performance," in *IEEE GLOBECOM Global Interne*, (Sydney, Australia), November 1998.

[45] Ben Abdesslem, F., Phillips, A., and Henderson, T., "Less is more: energy-efficient mobile sensing with senseless," in *Proceedings of ACM MobiHeld '09*, (Barcelona, Spain), 2009.

[46] Bharambe, A. R., Herley, C., and Padmanabhan, V. N., "Some observations on bittorrent performance," *SIGMETRICS Perform. Eval. Rev.*, 2005.

[47] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and Weiss, W., "An architecture for differentiated service," 1998.

[48] Braden, R., Clark, D., and Shenker, S., "Integrated services in the internet architecture: an overview," 1994.

[49] Brezmes, T., Gorricho, J.-L., and Cotrina, J., "Activity recognition from accelerometer data on a mobile phone," in *Proceedings of IWANN '09*, (Salamanca, Spain), 2009.

[50] Campbell, A. T., Eisenman, S. B., Fodor, K., Lane, N. D., Lu, H., Miluzzo, E., Musolesi, M., Peterson, R. A., and Zheng, X., "Transforming the social networking experience with sensing presence from mobile phones," in *Proceedings of ACM SenSys '08*, (Raleigh, NC, USA), 2008.

[51] Chan, M. C. and Woo, T. Y. C., "Cache-based compaction: A new technique for optimizing web transfer," in *Proceedings of IEEE INFOCOM '99*, (New York, NY), 1999.

[52] Chandra, S. and Vahdat, A., "Application-specific network management for energy-aware streaming of popular multimedia formats," in *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, (Berkeley, CA, USA), 2002.

[53] Chunlong Guo, L. C. Z. and Rabaey, J., "Low power distributed mac for ad hoc sensor radio networks," in *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, pp. 2944–2948, 2001.

[54] Constandache, I., Gaonkar, S., Sayler, M., Choudhury, R. R., and Cox, L., "Enloc: Energy-efficient localization for mobile phones," in *Proceedings of IEEE INFOCOM Mini Conference '09*, (Rio de Janeiro, Brazil), 2009.

[55] Conti, M., Gregori, E., and Turi, G., "A cross-layer optimization of gnutella for mobile ad hoc networks," in *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, (New York, NY, USA), 2005.

[56] Czerwinski, S. and Joseph, A., "Using simple remote evaluation to enable efficient application protocols in mobile environments," in *Proceedings of the 1st IEEE International Symposium on Network Computing and Applications*, (Cambridge, MA), 2001.

[57] de Lara, E., Wallach, D., and Zwaenepoel, W., "Puppeteer: component-based adaptation for mobile computing (poster session)," *SIGOPS Oper. Syst. Rev.*, vol. 34, no. 2, p. 40, 2000.

[58] de Lara, E., Wallach, D. S., and Zwaenepoel, W., "Hats: Hierarchical adaptive transmission scheduling," in *Multimedia Computing and Networking Conference (MMCN)*, (San Jose, California, USA), 2002.

[59] Dellarocas, C., "Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior," in *EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*, (New York, NY, USA), 2000.

[60] Dwyer, D. and Bharghavan, V., "A mobility-aware file system for partially connected operation,"

[61] Eisenman, S. B., Miluzzo, E., Lane, N. D., Peterson, R. A., Ahn, G.-S., and Campbell, A. T., "The bikenet mobile sensing system for cyclist experience mapping," in *Proceedings of ACM SenSys '07*, (Sydney, Australia), 2007.

[62] Fall, K., "Network emulation in the vint/ns simulator," in *ISCC '99: Proceedings of the The Fourth IEEE Symposium on Computers and Communications*, (Washington, DC, USA), 1999.

[63] Gaonkar, S., Li, J., Choudhury, R. R., Cox, L., and Schmidt, A., "Micro-blog: sharing and querying content through mobile phones and social participation," in *Proceedings of ACM MobiSys '08*, (Breckenridge, CO, USA), 2008.

[64] Gellersen, H. W., Schmidt, A., and Beigl, M., "Multi-sensor context-awareness in mobile devices and smart artifacts," *Mob. Netw. Appl.*, vol. 7, no. 5, pp. 341–351, 2002.

[65] Györbíró, N., Fábián, A., and Hományi, G., "An activity recognition system for mobile phones," *Mob. Netw. Appl.*, vol. 14, no. 1, pp. 82–91, 2009.

[66] Henderson, T. and Katz, R., "Transport protocols for Internet-compatible satellite networks," *IEEE Journal on Selected Areas in Communications(JSAC)*, vol. 17, pp. 345–359, Feb. 1999.

[67] Henderson, T., Kotz, D., and Abyzov, I., "The changing usage of a mature campus-wide wireless network," in *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, (New York, NY, USA), 2004.

[68] Hoh, B., Gruteser, M., Herring, R., Ban, J., Work, D., Herrera, J.-C., Bayen, A. M., Annavaram, M., and Jacobson, Q., "Virtual trip lines for distributed privacy-preserving traffic monitoring," in *Proceedings of ACM MobiSys '08*, (Breckenridge, CO, USA), 2008.

[69] Hsieh, H.-Y., Kim, K.-H., and Sivakumar, R., "On achieving weighted service differentiation: An end-to-end perspective," in *IEEE IWQoS '03: Proceedings of the International Workshop on Quality of Service*, (Monterey, CA, USA), 2003.

[70] Hsieh, H.-Y., Kim, K.-H., Zhu, Y., and Sivakumar, R., "A receiver-centric transport protocol for mobile hosts with heterogeneous wireless interfaces," in *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, (New York, NY, USA), pp. 1–15, ACM, 2003.

[71] Huang, C.-M., Hsu, T.-H., and Hsu, M.-F., "A file discovery control scheme for p2p file sharing applications in wireless mobile environments," in *ACSC '05: Proceedings of the Twenty-eighth Australasian conference on Computer Science*, pp. 39–47, 2005.

[72] Hull, B., Bychkovsky, V., Zhang, Y., Chen, K., Goraczko, M., Miu, A., Shih, E., Balakrishnan, H., and Madden, S., "Cartel: a distributed mobile sensor computing system," in *Proceedings of ACM SenSys '06*, (Boulder, Colorado, USA), 2006.

[73] IXIA. http://www.ixiacom.com/, (03/2010).

[74] Iyer, S., Rowstron, A., and Druschel, P., "Squirrel: a decentralized peer-to-peer web cache," in *Proceedings of PODC '02*, (Monterey, CA), 2002.

[75] Kang, S., Lee, J., Jang, H., Lee, H., Lee, Y., Park, S., Park, T., and Song, J., "Seemon: scalable and energy-efficient context monitoring framework for sensor-rich mobile environments," in *Proceedings of ACM MobiSys '08*, (Breckenridge, CO, USA), 2008.

[76] Korhonen, J. and Wang, Y., "Power-efficient streaming for mobile terminals," in *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*, (New York, NY, USA), 2005.

[77] Krashinsky, R. and Balakrishnan, H., "Minimizing energy for wireless web access with bounded slowdown," in *MobiCom '02*, (New York, NY, USA), 2002.

[78] Kuzmanovic, A. and Knightly, E. W., "Tcp-lp: low-priority service via end-point congestion control," *IEEE/ACM Trans. Netw.*, vol. 14, no. 4, pp. 739–752, 2006.

[79] Legout, A., Liogkas, N., Kohler, E., and Zhang, L., "Clustering and sharing incentives in bittorrent systems," in *SIGMETRICS '07: Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, (New York, NY, USA), 2007.

[80] Lester, J., Choudhury, T., Borriello, G., Consolvo, S., Landay, J., Everitt, K., and Smith, I., "Sensing and modeling activities to support physical fitness," in *Proceedings of UbiComp '05*, (Tokyo, Japan).

[81] Lian, Q., Zhang, Z., Yang, M., Zhao, B. Y., Dai, Y., and Li, X., "An empirical study of collusion behavior in the maze p2p file-sharing system," in *IEEE ICDCS '07: Proceedings of 27th IEEE International Conference on Distributed Computing Systems*, 2007.

[82] Lin, K., Kansal, A., Lymberopoulos, D., and Zhao, F., "Energy-accuracy aware localization for mobile devices," in *Proceedings of ACM MobiSys '10*, (San Francisco, California, USA).

[83] Linga, P., Gupta, I., and Birman, K., "A churn-resistant peer-to-peer web caching system," in *Proceedings of the 2003 ACM workshop on Survivable and self-regenerative systems*, (Fairfax, VA, USA), 2003.

[84] Lu, H., Pan, W., Lane, N. D., Choudhury, T., and Campbell, A. T., "Soundsense: scalable sound sensing for people-centric applications on mobile phones," in *Proceedings of ACM MobiSys '09*, (Kraków, Poland), 2009.

[85] Lu, L., Han, J., Hu, L., Huai, J., Liu, Y., and Ni, L. M., "Pseudo trust: Zero-knowledge based authentication in anonymous peer-to-peer protocols," in *IPDPS '07: Proceedings of the 21th IEEE International Parallel and Distributed Processing Symposium*, 2007.

[86] Miluzzo, E., Lane, N. D., Fodor, K., Peterson, R., Lu, H., Musolesi, M., Eisenman, S. B., Zheng, X., and Campbell, A. T., "Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application," in *Proceedings of ACM SenSys '08*, (Raleigh, NC, USA), 2008.

[87] Mogul, J. C., Chan, Y. M., and Kelly, T., "Design, implementation, and evaluation of duplicate transfer detection in http," in *Proceedings of NSDI'04*, (San Francisco, CA), 2004.

[88] Mohan, P., Padmanabhan, V. N., and Ramjee, R., "Nericell: rich monitoring of road and traffic conditions using mobile smartphones," in *Proceedings of ACM SenSys '08*, (Raleigh, NC, USA), 2008.

[89] Mohomed, I., Cai, J. C., Chavoshi, S., and de Lara, E., "Context-aware interactive content adaptation," in *MobiSys '06: Proceedings of the 4th international conference on Mobile systems, applications and services*, (New York, NY, USA), pp. 42–55, ACM, 2006.

[90] Noble, B. D., Satyanarayanan, M., Narayanan, D., Tilton, J. E., Flinn, J., and Walker, K. R., "Agile application-aware adaptation for mobility," in *Proceedings of the 16th ACM Symposium on Operating System Principles*, (Saint Malo, France), 1997.

[91] Paek, J., Kim, J., and Govindan, R., "Energy-efficient rate-adaptive gps-based positioning for smartphones," in *Proceedings of ACM MobiSys '10*, (San Francisco, California, USA).

[92] Pahdye, J. and Floyd, S., "On inferring tcp behavior," in *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, (New York, NY, USA), 2001.

[93] Pallis, G. and Vakali, A., "Insight and perspectives for content delivery networks," *Commun. ACM*, vol. 49, no. 1, pp. 101–106, 2006.

[94] Park, K. and Pai, V. S., "Scale and performance in the coblitz large-file distribution service," in *Proceedings of NSDI'06*, (San Jose, CA), 2006.

[95] Parvez, N., Williamson, C., Mahanti, A., and Carlsson, N., "Analysis of bittorrent-like protocols for on-demand stored media streaming," *SIGMETRICS Perform. Eval. Rev.*, 2008.

[96] Paul, S., Ayanoglu, E., Porta, T. F. L., Chen, K.-W. H., Sabnani, K. E., and Gitlin, R. D., "An asymmetric protocol for digital cellular communications," in *INFOCOM '95: Proceedings of the Fourteenth Annual Joint Conference of the IEEE Computer and Communication Societies (Vol. 3)-Volume*, (Washington, DC, USA), p. 1053, IEEE Computer Society, 1995.

[97] Perkins, C., "Mobile IP," *IEEE Communications Magazine*, vol. 40, pp. 66–82, May 2002.

[98] Qiu, D. and Srikant, R., "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, (New York, NY, USA), 2004.

[99] Rabin, M. O., "Fingerprinting by random polynomials," *Technical Report TR-15-81,Center for Research in Computer Technology*, 1981.

[100] Rahmati, A. and Zhong, L., "Context-for-wireless: context-sensitive energy-efficient wireless data transfer," in *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, (New York, NY, USA), pp. 165–178, ACM, 2007.

[101] Rapier, C. and Bennett, B., "High speed bulk data transfer using the ssh protocol," in *MG '08: Proceedings of the 15th ACM Mardi Gras conference*, (New York, NY, USA), pp. 1–7, ACM, 2008.

[102] Rhea, S. C., Liang, K., and Brewer, E., "Value-based web caching," in *Proceedings of WWW '03*, (Budapest, Hungary), 2003.

[103] Rodriguez, P., Mukherjee, S., and Rangarajan, S., "Session level techniques for improving web browsing performance on wireless links," in *WWW '04: Proceedings of the 13th international conference on World Wide Web*, (New York, NY, USA), pp. 121–130, ACM, 2004.

[104] Saleh, O. and Hefeeda, M., "Modeling and caching of peer-to-peer traffic," in *Proceedings of ICNP '06*, (Washington, DC), 2006.

[105] Santos, J. and Wetherall, D., "Increasing effective link bandwidth by suppressing replicated data," in *Proceedings of ATEC '98*, (New Orleans, LA, USA), 1998.

[106] Satyanarayanan, M., Kistler, J. J., Kumar, P., Okasaki, M. E., Siegel, E. H., and Steere, D. C., "Coda: A highly available file system for a distributed workstation environment," *IEEE Transactions on Computers*, vol. 39, no. 4, pp. 447–459, 1990.

[107] Shih, E., Bahl, P., and Sinclair, M. J., "Wake on wireless: an event driven energy saving strategy for battery operated devices," in *Proceedings of ACM MobiCom '02*, (Atlanta, Georgia, USA), 2002.

[108] Sinha, P., Venkitaraman, N., Sivakumar, R., and Bharghavan, V., "Wtcp: a reliable transport protocol for wireless wide-area networks," in *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, (New York, NY, USA), pp. 231–241, ACM, 1999.

[109] Snoeren, A. and Balakrishnan, H., "An end-to-end approach to host mobility," in *MOBICOM*, (Boston, MA, USA), Aug. 2000.

[110] Sorber, J., Banerjee, N., Corner, M. D., and Rollins, S., "Turducken: hierarchical power management for mobile devices," in *Proceedings of ACM MobiSys '05*, (Seattle, Washington), 2005.

[111] Spring, N. T. and Wetherall, D., "A protocol-independent technique for eliminating redundant network traffic," *SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 4, pp. 87–95, 2000.

[112] Srivatsa, M., Xiong, L., and Liu, L., "Trustguard: countering vulnerabilities in reputation management for decentralized overlay networks," in *WWW '05: Proceedings of the 14th international conference on World Wide Web*, (New York, NY, USA), 2005.

[113] Su, A.-J., Choffnes, D. R., Kuzmanovic, A., and Bustamante, F. E., "Drafting behind akamai (travelocity-based detouring)," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 435–446, 2006.

[114] The Network Simulator, "ns-2." http://www.isi.edu/nsnam/ns, (03/2010).

[115] Viredaz, M. A., Brakmo, L. S., and Hamburgen, W. R., "Energy management on handheld devices," *ACM Queue*, vol. 1, no. 7, pp. 44–52, 2003.

[116] Walfish, M., Balakrishnan, H., Karger, D., and Shenker, S., "Dos: Fighting fire with fire," in *Proceedings of the 4th ACM Workshop on Hot Topics in Networks (HotNets)*, (College Park, MD), 2005.

[117] Wang, Y., Lin, J., Annavaram, M., Jacobson, Q. A., Hong, J., Krishnamachari, B., and Sadeh, N., "A framework of energy efficient mobile sensing for automatic user state recognition," in *Proceedings of ACM MobiSys '09*, (Kraków, Poland), 2009.

[118] Wei, K., Chen, Y.-F., Smith, A. J., and Vo, B., "Whopay: a scalable and anonymous payment system for peer-to-peer environments," in *IEEE ICDCS '06: Proceedings of 27th IEEE International Conference on Distributed Computing Systems*, 2006.

[119] Wei Ye, John Heidemann, D. E., "An energy-efficient mac protocol for wireless sensor networks," in *INFOCOM*, 2002.

[120] Yan, H., Krishnan, R., Watterson, S. A., and Lowenthal, D. K., "Client-centered energy savings for concurrent http connections," in *NOSSDAV '04: Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video*, (New York, NY, USA), 2004.

[121] Yoon, J., Noble, B., and Liu, M., "Surface street traffic estimation," in *Proceedings of ACM MobiSys '07*, (San Juan, Puerto Rico), 2007.

[122] Zandy, V. C. and Miller, B. P., "Reliable network connections," in *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, (New York, NY, USA), 2002.

[123] Zhang, Z., Chen, S., and Yoon, M., "March: A distributed incentive scheme for peer-to-peer networks," in *IEEE INFOCOM '07: Proceedings of the 26th IEEE International Conference on Computer Communications*, (Anchorage, Alaska, USA), 2007.

[124] Zhu, H. and Cao, G., "On supporting power-efficient streaming applications in wireless environments," *IEEE Transactions on Mobile Computing*, 2005.

[125] Ziv, J. and Lempel, A., "A universal algorithm for sequential data compression," *Information Theory, IEEE Transactions on*, vol. 23, pp. 337–343, May 1977.

[126] Ziv, J. and Lempel, A., "Compression of individual sequences via variable-rate coding," *Information Theory, IEEE Transactions on*, vol. 24, pp. 530–536, Sep 1978.

VITA

Zhenyun Zhuang was born in Shandong Province, China. He received his B.E. degree in Information Engineering from Beijing University of Posts and Telecommunications in 1997, and his M.S. degree from the Tsinghua University in 2002. He worked as an engineer in China from 1997 to 1999. He joined the Ph.D. program of College of Computing at Georgia Tech in Fall 2004, and has been a Research Assistant in Prof. Sivakumar's GNAN research group.