2-2011

Architecting Protocols to Enable Mobile Applications in Diverse Wireless Networks

Aruna Balasubramanian
University of Massachusetts Amherst

ARCHITECTING PROTOCOLS TO ENABLE MOBILE APPLICATIONS IN DIVERSE WIRELESS NETWORKS

A Dissertation Presented

by

ARUNA BALASUBRAMANIAN

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2011

Computer Science

ARCHITECTING PROTOCOLS TO ENABLE MOBILE APPLICATIONS IN DIVERSE WIRELESS NETWORKS

A Dissertation Presented

by

ARUNA BALASUBRAMANIAN

Approved as to style and content by:

Arun Venkataramani, Co-chair

Brian Neil Levine, Co-chair

Donald Towsley, Member

Lixin Gao, Member

Ratul Mahajan, Member

Andrew G. Barto, Department Chair
Computer Science

ACKNOWLEDGMENTS

I want to express my deepest gratitude to my advisors Brian Levine and Arun Venkataramani. They are the best advisors one can get, in my (very biased) opinion. I thank them for encouraging me to always aim high, for spending countless hours with me discussing research problems and ideas, for their feedback on my writing, for giving me confidence when I needed it the most, and for being excellent role models. Their love for research and intellectual curiosity has rubbed off on me, and I hope to never lose sight of it in my research career.

I thank Ratul Mahajan, whom I had the good fortune of working with on two of the four problems described in this thesis. It was great fun working with Ratul, and he has the uncanny ability of teaching research skills without ever appearing to do so. I thank him for always treating me as a colleague even when I was a new graduate student.

I thank Don Towsley for his thorough comments on my thesis and for asking tough questions. His criticisms went a long way in improving this thesis. I thank Lixin Gao for her feedback and comments that was especially useful in making the motivation of my work stronger.

I thank DARPA, NSF, and Microsoft Research (MSR) for funding my graduate studies.

I thank Victor Bahl, one of my mentors and well-wishers whose advise I greatly value. His vision and breath of knowledge is an inspiration to all. I thank my collaborators and mentors at MSR including Ranveer Chandra, Jitu Padhye, Stefan Saroui, and Alec Wolman. I thank my fellow interns at MSR for the many many

passionate discussions and arguments, both about research and otherwise. I especially thank Eduardo Cuervo, Murtaza Motiwala, Rohan Murty, and Ramya Raghavendra. A special shout out to the 112/3001 intern mafia for making memorable what was possibly one of my most challenging summers.

I thank the members of the PRISMS and the LASS lab for the many research discussions, feedback on presentations, and general camaraderie. In particular, I thank Nilanjan Banerjee, Jacob Sorber, Ming Li, Bruno Ribeiro, Hamed Saroush, and Xiaozheng Tie. I thank Brian Lynn for helping me a great deal when deploying experiments on DieselNet; without his help this thesis would have taken much longer to complete.

I thank everyone in the computer science department at UMass, one of the most friendly and collaborative departments I have known. I thank the office staff and all of the secretaries for cheerfully helping me out with administrative tasks. I thank CSCF for making sure that I did not have to worry about equipment problems in the last 5 years.

I have made some invaluable friends in the last five years. I thank them for providing me with the much needed distraction and for making me feel part of a large family. I thank Gal Niv, Yariv Levy, David Cooper, Erin Cooper, Tim Wood, Megan Olsen, Shiraj Sen, and many more. It will be impossible to list them all.

I started my graduate studies at Buffalo, and my stint in Buffalo gave me the confidence and desire to pursue a Ph.D. I thank R. Sridhar and Sumita Mishra for introducing me to the research process. I thank Hung Ngo, whose algorithms class made me realize my love for science. I thank my friends in Buffalo, especially the members of the 215 Furnas lab, for their amazing company and for the all round fun they provided. I thank Karthik Thyagarjan for helping me out every time I had an optimization question, but most of all for being a wonderful friend.

I am greatly indebted to my family; without them I will not be where I am today. I give my heartfelt thanks to them—to my father, for instilling in me a love for science, to my mother, for giving me the courage and independence to pursue what I wanted to, to my brother, for his support no matter what I decided to do, and to my grandmother, for having immense (and often unwarranted) belief in my abilities.

Most of all, I thank my husband Niranjan, for this thesis would not be possible without him. I thank him for encouraging me to pursue a Ph.D, and for never letting me take the easy way out. I thank him for having more confidence in me than I had in myself. I thank him for his quiet support when I was going through stressful times. But above all, I thank him for making my life infinitely more fun than I ever thought possible.

ABSTRACT

ARCHITECTING PROTOCOLS TO ENABLE MOBILE APPLICATIONS IN DIVERSE WIRELESS NETWORKS

FEBRUARY 2011

ARUNA BALASUBRAMANIAN

M.S., UNIVERSITY OF BUFFALO, SUNY

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Arun Venkataramani and Professor Brian Neil Levine

The goal of this thesis is to architect robust protocols that overcome disruptions and enable applications in diverse mobile networks. Mobile users operate in diverse environments, starting from mostly connected cellular networks to mostly disconnected delay tolerant networks (DTNs). Each of these networks are prone to frequent disruptions due to mobility, coverage holes, poor channel conditions, and other factors. Designing protocols to tolerate such disruptions is challenging because of the extreme uncertainty in mobile wireless environments. In this thesis, I focus on four networks that span the diverse connectivity spectrum and answer the following questions for each network: (1) What are the disruption characteristics in the network and what are the opportunities that can be exploited in the network?; and (2) What protocol design best exploits the opportunities to overcome disruptions and enable the specific application?

In this thesis, the key insight used to tolerate disruptions is opportunistic resource usage. Opportunistic mechanisms use resources as they become available and are therefore naturally resilient to uncertainty. Specifically, I present four protocols that overcome disruptions and enable applications in diverse networks: 1) RAPID, which uses opportunistic replication to enable bulk transfer in mostly disconnected networks; 2) Thedu, which uses opportunistic prefetching to enable web search in intermittently connected networks; 3) ViFi, which uses opportunistic forwarding to enable Voice over IP (VoIP) in mostly connected mesh networks; and 4) Wiffler, which uses opportunistic augmentation to improve application performance in mostly connected cellular networks. The naive use of opportunism can waste resources and hurt performance. I show how, in most cases, utility-driven protocols can be used to implement opportunism in resource-constrained wireless environments.

Finally, I present a detailed evaluation of the protocols using implementation and deployment experiments on two large scale vehicular testbeds. Deployment on a real testbed shows that the protocols are practical and can be implemented in realistic usage environments. The evaluations show that the protocols signicantly improve performance of applications compared to the state-of-the-art, in their respective environments.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

The vision for this thesis is: *Providing reliable network connectivity to mobile users in diverse environments.* In recent years, there has been a prolific increase in network-enabled mobile devices. In parallel, there has been an increase in infrastructure support for mobile access—planned support in the form of cellular and WiFi mesh deployments [1, 33], and ad hoc support in the form of vehicular networks [96]. However, mobile access is nowhere close to being ubiquitous because the network is extremely unreliable and prone to frequent disruptions. For example, 3G users in big cities experience frequent disruptions [115] even though cellular infrastructure is supposed to provide ubiquitous coverage.

Designing protocols that overcome disruptions in mobile environments is challenging because of two reasons : *(i) Uncertainty in network conditions:* The uncertainty stems from mobility, frequent topology changes, and fluctuating channel conditions. As a result, mobile protocols need to make design decisions based on partial or incorrect knowledge about the environment, and, *(ii) Network diversity:* Mobile users require network access in diverse environments, starting from mostly connected (for example, in the city center) to mostly disconnected (for example, in a subway). The environments are prone to varying disruption characteristics and present unique challenges. Designers need to first uncover the challenges specific to the network, and then design protocols that can address the challenges.

1.1 Thesis statement

The goal of my work is to architect robust protocols that overcome disruptions in diverse environments. To this end, this dissertation seeks to establish the following thesis: *Disruptions in diverse mobile environments can be overcome by exploiting resources opportunistically, often using utility-driven protocols.*

1.2 Thesis overview

In support for my thesis, I designed a suite of opportunistic protocols that overcome disruptions in mobile wireless networks. Opportunistic protocols use resources as they become available rather than planning for them *a priori*. As a result, the protocols easily adapt to changing network conditions, making them resilient to uncertainty.

There are two aspects to the problems studied in this thesis; first is diversity in network connectivity and second is application interactivity. Figure 1.1 illustrates these two dimensions. In highly disconnected environments, the goal of this thesis is to design protocols that can support bulk transfer applications that are non-interactive. Clearly, in disconnected environments where nodes are only connected for a short duration, interactive applications such as web search or web browsing cannot be supported. However, as the connectivity increases, the goal of this thesis is to design protocols that can overcome disruptions to support increasingly interactive applications.

Specifically, I focus on four networks and corresponding applications that span the two dimensions of network connectivity and application interactivity. I then answer the following questions:

- *What are the disruption characteristics in the network and what are the opportunities that can be exploited in the network?*

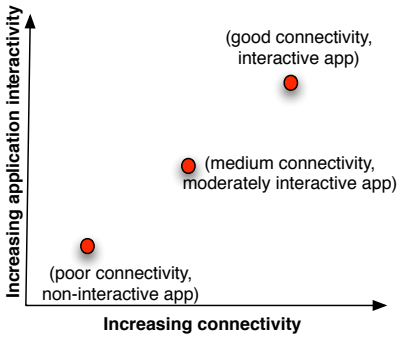


Figure 1.1. Introduction: Dimensions of the problem studied in this thesis - Network connectivity and Application interactivity.

- *What protocol design best exploits the opportunities to overcome disruptions and enable the given application?*

Below, I present an overview of the four networks and applications and the methodology I follow to answer the research questions. In Chapter 2, I present a more detailed description of the network environments and the research methodology, and I place both in context of related research work.

1.2.1 Diverse networks considered in this thesis

Figure 1.2 is a qualitative illustration of the specific network environment considered in this work: Mostly disconnected, Intermittently connected, Mostly connected mesh, and Mostly connected cellular network. The networks are presented in terms of the connection and disconnection duration. The disconnection duration is the period of time when a mobile user has no contact with the infrastructure. The connection duration is the contiguous period of time when a mobile node is in contact with infrastructure. We define infrastructure to be a cellular tower, a WiFi access point (AP), or another network node.

Mostly disconnected networks, the outer region, refers to challenged networks where infrastructure is minimal and connectivity is infrequent. Such networks are

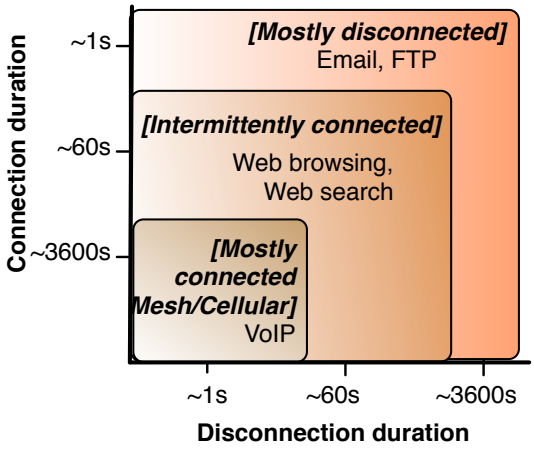


Figure 1.2. Introduction: The diverse wireless environment considered in this thesis.

also known as disruption tolerant networks (DTNs). Disconnection durations are typically on the order of a few hours, and connection durations are on the order of a few minutes/seconds. DTNs are most useful in scenarios where infrastructure is expensive or difficult to deploy, such as wildlife monitoring, disaster relief, and in rural or developing regions. The goal of this thesis is to support non-interactive applications such as email in the DTN environment. Supporting non-interactive applications is challenging in this environment because there may be no contemporaneous end-to-end path and traditional routing and transport protocols break down.

Intermittently connected, the middle region, refers to networks where disconnection durations are smaller, on the order of a few minutes. For example, a mobile node connected to a WiFi AP in a coffee shop only has intermittent access. The node gets disconnected when it moves out of range of the AP, and is not connected until it is in range of another AP. The disconnection duration is shorter than DTN environments, but the environment is still not well connected. Such intermittent access, if harnessed, can be used for a variety of applications such as mobile advertisement, local recommendations, and road monitoring. The goal of this thesis is to support a

moderately interactive web search application in this environment. Web search cannot tolerate even a few minutes of disruption, and therefore cannot be supported *as is* in an intermittently connected environment.

Mostly connected, the inner region, refers to networks that have very short disconnection durations. Mostly connected networks can be classified into cellular network or mesh networks, depending on whether they are supported by a cell tower or a WiFi mesh infrastructure, respectively. The infrastructure is planned so that it provides overlapping coverage to mobile users. Even in such well provisioned environments, supporting highly interactive applications such as Voice over IP (VoIP) is difficult because of short disruptions in connectivity, either due to poor channel conditions [22, 76] or network overload [21].

The key objective of this thesis is to design protocols that enable the applications corresponding to the different regions, as shown in Figure 1.2. Today, these applications can not be supported, and even if supported, suffer from poor performance [17, 22, 19, 21].

1.2.2 Research Methodology

The research methodology used in this thesis is comprised of the following three components: *(i)* Uncovering challenges and opportunities in the environment using testbed-based measurements; *(ii)* Addressing the challenges by designing protocols that leverage opportunism; and *(iii)* Evaluating the protocols using implementation and testbed deployment.

I use two large-scale vehicular testbeds, Dome [7, 101] and VanLan [110], to conduct measurement studies in the four networks. The measurement studies help uncover problems and opportunities in the network that may be otherwise unnoticed. I leverage these measurements to design protocols that improve application performance. For example, in ViFi (Chapter 5), we use measurements on VanLAN to show

that several APs overhear a dropped packet in a mesh. Based on this observation, we design a protocol that leverages AP overhearing to reduce disruptions and enable interactive applications in the mesh environment.

The key insight underlying the approaches proposed in this thesis is *opportunistic resource usage*. I leverage four opportunistic mechanisms: *Replication, Aggressive prefetching, Opportunistic forwarding, and Opportunistic augmentation*, to reduce disruptions in the four networks environments. Opportunistic mechanisms use resources as they become available and are naturally resilient to uncertainty. However, naive use of opportunism can increase resource overhead and hurt performance. My contribution is a set of utility-driven protocols that implement the opportunistic mechanisms in a resource constrained environment.

Finally, evaluating protocols designed for the mobile wireless environment is difficult because analytical models and simulation tools do not accurately capture aspects of the wireless channel [69]. I evaluate the protocols developed in this thesis using implementation and deployment, primarily on the Dome [7, 101] testbed. I evaluate mesh protocols in the VanLAN [110] mesh testbed. The deployment-based evaluation takes into account aspects of the wireless channel that are difficult to capture using analysis or simulation tools and shows that the protocols can be deployed in realistic usage environments. For a broader evaluation across a range of environmental factors, I supplement the deployment-based evaluation with trace-driven simulations using real-world traces.

1.3 Thesis Outline

The rest of this thesis is organized as follows: In Chapter 2, I present a detailed overview of the network environments considered in this thesis, the opportunistic mechanisms leveraged, and the mobile testbeds used for evaluation.

In Chapter 3, I describe RAPID, a DTN protocol that reliably routes packets under extreme uncertainty. A RAPID node *opportunistically replicates* packets through multiple paths to increase the probability that the packet will be delivered. I first present a measurement study on the Dome testbed that shows that while replication can improve performance in DTNs, naive replication hurts performance when resources are limited. To address the resource management challenge, RAPID uses a utility-driven protocol that tunes replication according to the resource availability; during a short connection opportunity, RAPID replicates packets such that the marginal increase in the system's utility justifies the resources consumed. I present a detailed evaluation of RAPID using deployment-experiments on Dome and using trace-driven simulations. A paper describing RAPID was published at SIGCOMM 2007 [17], and a full version appeared in the IEEE/ACM Transactions on Networking 2010 [20].

In Chapter 4, I describe Thedu, a protocol that overcomes disruptions to enable web search in intermittently connected networks. I first present a measurement study conducted on the Dome testbed to understand the challenges in supporting web search in intermittently connected networks. I then describe Thedu, a protocol that leverages *aggressive prefetching* to transform the interactive web search process to a transactional process. Thedu uses a combination of Information Retrieval (IR) techniques and utility-driven prioritization to ensure that the most useful web pages are prefetched during the short bandwidth opportunity. Finally, I present an evaluation of Thedu using deployment experiments on Dome and using trace-driven simulations. A paper describing Thedu was published at CHANTS workshop [24], and a full version was published at MobiCom 2008 [19].

In Chapter 5, I describe ViFi, a protocol that enables interactive applications in mostly connected WiFi meshes. I present a measurement study on the VanLAN mesh testbed that shows that several APs overhear a dropped packet because of the broadcast nature of the wireless medium. Next, I describe ViFi, a protocol that lever-

ages opportunistic overhearing by allowing an AP that overhears a dropped packet to forward the packet. ViFi uses a utility-driven sender prioritization to coordinate among APs such that the forwarding reduces disruptions, while minimizing wasted transmissions. Finally, I present an evaluation of ViFi using deployment experiments on VanLAN and trace-driven simulations using Dome traces. A paper describing ViFi was published at SIGCOMM 2008 [22].

In Chapter 6, I describe Wiffler, a protocol that augments 3G networks using WiFi. I present a detailed measurement study of the availability of 3G and WiFi networks in 3 cities that shows that WiFi is available only a fraction of the time that 3G is available and has poorer performance compared to 3G. I then describe Wiffler, a protocol that overcomes the availability and performance challenges in WiFi to augment the overloaded 3G network. Wiffler predicts future WiFi availability to maximize the data offloaded to the WiFi network, while not affecting application performance. Finally, I present an evaluation of Wiffler using deployment on Dome and using trace-driven simulations based on traces collected in two cities. A paper describing Wiffler was published at MobiSys 2010 [21].

In Chapter 7, I describe potential avenues of future research and Chapter 8 concludes this thesis.

CHAPTER 2

BACKGROUND

In this section, I provide context to the problems addressed in this thesis by elaborating on three topics: diverse wireless networks, testbed deployment, and opportunistic mechanisms.

In Section 2.1, I describe the four wireless networks considered in this thesis, including real-world instantiations of the networks and the challenges faced in the networks. In Section 2.2, I describe the Dome and VanLAN testbeds used in this thesis to conduct measurement studies and to evaluate protocols. I will also present a survey of related testbed deployments. Finally, in Section 2.3, I elaborate on the opportunistic mechanisms leveraged in this thesis and describe related research that have also employed these mechanisms to solve different problems.

2.1 Diverse Wireless Networks: Applications and Challenges

Below, I describe in detail the characteristics of four diverse wireless networks (see Figure 1.2): Mostly disconnected DTNs, Intermittently connected networks, Well connected meshes, and Well connected cellular networks.

2.1.1 Mostly Disconnected Networks

Mostly Disconnected Networks, also known as DTNs, are networks that are formed in extremely challenged environments where there is no end-to-end connectivity between nodes. DTNs were first envisioned for inter-planetary communication [47], where satellites communicate with each other when they come within range. The

Figure 2.1. Background: A solar-powered tracking device used in TurtleNet.

satellite nodes communicate using a *store-and-forward* technique, where a node stores packets and forwards it only when it is within range of another node.

The applicability of DTNs have since been extended to several other scenarios including wildlife monitoring, disaster relief, and rural and developing regions. In general, the DTN communication paradigm is most useful for environments where infrastructure is expensive to deploy or difficult to deploy. Below, I present two real world examples of the application of DTNs.

TurtleNet

TurtlNet [100, 9] is a mobile network deployed in Mississippi by researchers at UMass, in collaboration with biologists at the University of Southern Mississippi. The deployment consists of 17 tracking devices attached to Gopher Tortoises (Gopherus polyphemus), shown in Figure 2.1 and two GPRS-enabled base station deployed at the two ends of the swamp.

During operation, the devices log connection opportunities and record periodic sensor reading. The nodes in TurtleNet rarely have an end-to-end connection to one of the two deployed base stations, and therefore cannot transfer the logs directly. Instead, nodes use DTN routing: When two TurtleNet nodes come within communication range of each other, they exchange their logs. The logs are forwarded during subsequent node meetings until it is eventually delivered to the base station.

Traditionally, the logs are collected by trapping the animals or using manual radio telemetry, both of which are labor intensive and yield few data points. Alternatively, more base stations can be deployed to collect logs directly, but this is expensive and in some cases impossible. In contrast, DTNs allow for streaming of collected data to scientists without expensive infrastructure or manual intervention by routing data during opportunistic contacts. ZebraNet [117], is another example of a sensor deployment used for monitoring zebras in Kenya. ZebraNet also uses a DTN communication paradigm to collect data from the sensor nodes.

Kiosknet

The KioskNet system [96], developed at the University of Waterloo, provides low cost Internet access to rural areas by leveraging DTN communication. The KioskNet system is currently deployed in Southern India and in Ghana. Rural kiosks that are already deployed for connectivity are often unreliable due to failures in the telephone system.

Instead, the KioskNet project augments the telephone-based network using DTNs. The DTN is formed among vehicles equipped with an on-board computer and an external antenna that carry Internet traffic between the kiosks and the Internet gateway in the cities. Without DTN routing, such a connectivity would be possible only with extensive deployment of cellular or other infrastructure, which can be expensive.

Despite several real world applications for DTNs, many challenges need to be addressed for DTN communication to be robust and efficient. First, there is no end-to-end path between source and destination; existing wired and wireless routing protocols assume the presence of an end-to-end path, and as a result, break down in the DTN environment [17]. Second, making it difficult to determine the best next hop for routing because of the disconnected nature of the network and since nodes need to make decisions only with partial information. In this thesis, I present RAPID,

a routing protocol that overcomes the uncertainty in topology and the lack of end-to-end path in DTNs to reliably deliver packets.

2.1.2 Intermittently Connected Networks

There is increasing number of WiFi APs deployed in college campuses, airports, corporate campuses, and coffee shops [40]. A mobile user can get intermittent Internet access when she is within range of one of the APs. Such intermittent access, if harnessed, can be used for a variety of applications such as mobile advertisement, traffic management, and road monitoring. For example, Pothole Patrol [48] is a project that enables road monitoring in the real world, where vehicles gather data about road conditions and relay the data to a central repository during intermittent Internet access.

Several earlier studies in this topic characterize the throughput available to moving vehicles as they pass a WiFi AP. The Drive-Thru-Internet [87] project shows that communication between a vehicle and an AP is feasible even at speeds of 180Km/h. Hadaller et al. [58] perform a detailed study of TCP performance between a moving vehicle and an AP. The authors introduce several optimizations to TCP to improve throughput from moving vehicles.

There has been recent studies that focus on vehicle-AP interactions through a typical day, as the vehicle comes in contact with several organically deployed APs. The CarTel study [61, 49] shows that a vehicle can get an average throughput of 86Kb/s per day using open APs deployed in the Boston area. The MobTorrent [40] study shows that vehicles have a mean capacity of 4.5MB per AP meeting in a campus setting in Singapore. Our own work [19] in the Dome testbed shows that vehicles can transfer data to an AP every 8 minutes on average. Figure 2.2 shows the number of vehicle-AP meetings per day in Dome. For example, in the center grid in the Dome

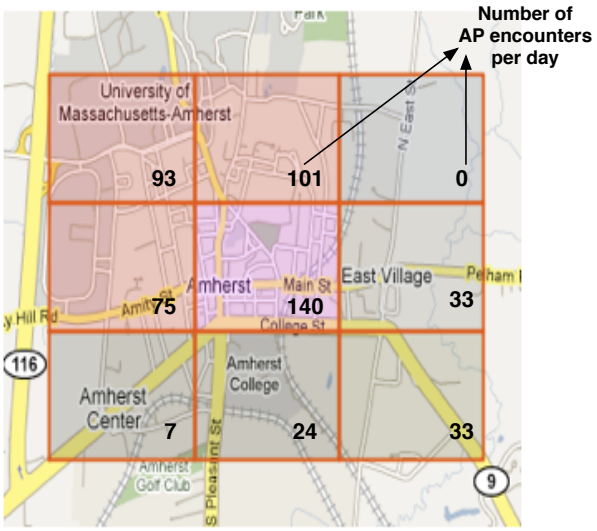


Figure 2.2. Background: Number of AP meetings in a 2 sq.mile area around the center of Amherst.

deployment region, corresponding to the town center in Amherst, a vehicle encounters APs 140 times a day.

The above studies focus on the available throughput for vehicular users. The throughput measure is appropriate for bulk-transfer applications, but not for interactive applications such as web search, where response time is of critical importance. It is unclear if interactive applications can be supported when connectivity is intermittent and unpredictable. In this thesis, I present a system called Thedu that adapts the interactive web search process to intermittently connected networks.

2.1.3 Mostly Connected Mesh

Recently, there has also been an increasing number of planned mesh deployments, and over 1,000 cities worldwide have deployed mesh networks [53]. A mesh network typically consists of a large number of wireless APs and a few gateway routers connected to the Internet. A few of the APs are directly connected to the gateway

routers, and the remaining APs route data to the gateway wirelessly through other mesh APs. The mesh network is deployed to provide continous coverage to users.

There are several real-world examples of mesh networks that serve entire communities or cities. GoogleWiFi [1] is a mesh network of over 500 WiFi nodes and 3 gateways that covers 95% of the city of Mountain View. In 2009, over 500 GB of data was sent through the GoogleWiFi mesh per day. Similarly, the TFA network in Houston [33] is a mesh network consisting of 18 backhaul nodes deployed in a single family residential neighborhood. The mesh testbed serves over 4,000 community residents. In Amherst, we have deployed a mesh network consisting of 26 Cisco APs that are mounted on light poles and buildings. On average, there were 86,838 HTTP connections (68% of the connections) to the mesh nodes from users per day.

WiFi meshes have the potential to provide low-cost and ubiquitous access to users, and it is a one time investment. For example, the small municipality in Amherst cannot afford 3G subscriptions for every worker; however, a one-time purchase of WiFi equipment is feasible.

For vehicular and mobile users, the primary challenge in supporting applications using a mesh network is in designing effective handoff policies. Because of uncertainty in channel conditions, it is challenging to make optimal handoff decisions as vehicles move from one mesh node to another. In this thesis, I present an effective handoff protocol called ViFi that enables highly interactive applications in vehicular-mesh networks.

2.1.4 Mostly Connected Cellular

Unlike WiFi, 3G cellular networks are designed for mobility, and can support a wide range of applications at vehicular speeds. They are deployed by commercial vendors and are fundamentally different from the organic WiFi networks in two ways: *(i)* They require massive cell tower infrastructure investment that is recovered through

per-month user subscriptions; and *(ii)* access by a client is strictly scheduled and requires tight integration with the physical layer. This ability requires expensive hardware at the client, but also provides better performance guarantees compared to WiFi networks that only support random access. In other words, while 3G networks are more expensive than WiFi networks, they provide consistent performance, can support ubiquitous access, and as a result, have been adopted by millions of users in urban areas [12].

Earlier generations of cellular packet-switched networks such as GPRS and EDGE networks (also called 2.5G) only supported a bandwidth of about 100Kbps, which is significantly low compared to what is possible from WiFi networks. However, 3G networks, for example UMTS-HSPA and WCDMA, support bandwidths comparable to WiFi. Currently, the cellular industry is working on the newest 4G standard called LTE (Long Term Evolution), that is projected to support extremely high throughputs in the order of 100 Mbps. These developments make cellular networks extremely attractive to urban users.

The challenge is that the popularity of 3G networks is creating immense pressure on the limited spectrum. Currently, commercial 3G networks are allocated 409.5 MHz of spectrum [13], and only about 50MHz of spectrum is available for future use. The FCC ruling on the use of unused TV spectrum (called whitespaces) can potentially increase the spectrum availability by at most 100-200 MHz [10]. However, the projected demand for spectrum by 2016 is 800-1000 MHz [13]. It is unclear how cellular providers can satisfy this demand. In this thesis, I present Wiffler, a system that augments 3G spectrum using cheaper WiFi connectivity.

2.2 Mobile testbeds

In this thesis, I use two vehicular testbeds to conduct measurement studies and to evaluate protocols in a realistic usage environment: Dome [8] and VanLAN [110].

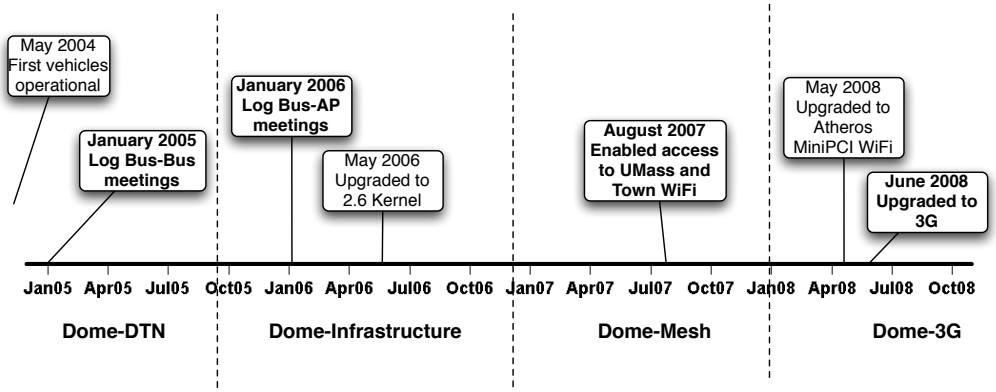


Figure 2.3. Background: Evolution of the Dome testbed.

I will describe the two testbeds in detail, and present a survey of related mobile testbeds.

2.2.1 Dome

Dome is a diverse testbed that allows us to experiment on different network environments, starting from DTNs to cellular networks. The testbed has been operational since 2005 and has evolved to include several different wireless networks. Figure 2.3 shows the evolution of Dome.

The testbed is composed of 40 buses operated by the Pioneer Valley Transport Authority (PVTA) at UMass. Each bus is fitted with off-the-shelf hardware as shown in Figure 2.4 consisting of a Hacom OpenBrick 1GHz Intel systems (referred to as bricks), a GPS receiver, 802.11abg mini PCI cards, 802.11g wireless access point, amd Wireless 3G USB modems. The buses service an areas of 150 sq.mile around the UMass campus. The bricks run a software module called LiveIP that scans for SSIDs, establishes and maintains WiFi connections, and informs applications of the state of the WiFi link.

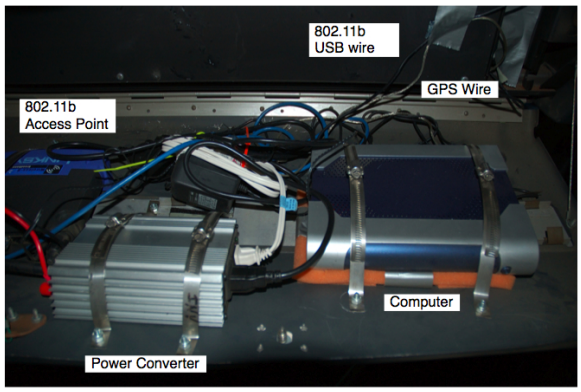


Figure 2.4. Background: The hardware on the Dome buses.

The computers on the buses collect logs of experiments and upload the logs to a central repository periodically. We use the experiment logs to evaluate protocols. Below, I describe the different wireless networks that Dome is comprised of.

Dome-DTN

Dome was originally deployed as a DTN testbed, where buses communicated with each other when they were in range, forming a sparse network. An 802.11b AP attached to each bus provides DHCP access to the bus, and a second USB-based 802.11b interface scans for other buses. Once another bus is within range, a connection is created to the remote AP using the LiveIP software. The buses send bulk data to each over TCP until they are disconnected. The total data transfered between two communicating buses is a function of several effects that are not perfectly modeled by simulations, including the effect of the wireless channel, the speeds of the buses, and other events such as traffic and signal.

In this thesis, I evaluate RAPID (Chapter 3) using the Dome-DTN testbed.

Dome-Infrastructure

In 2006, the Dome-DTN testbed was upgraded to the Dome-Infrastructure testbed, where the buses interacted with APs. To this end, the LiveIP software scanned not only for other buses, but also for open APs. If multiple APs were found in a scan, LiveIP attempted to associate to an AP with the best signal strength. After associa-

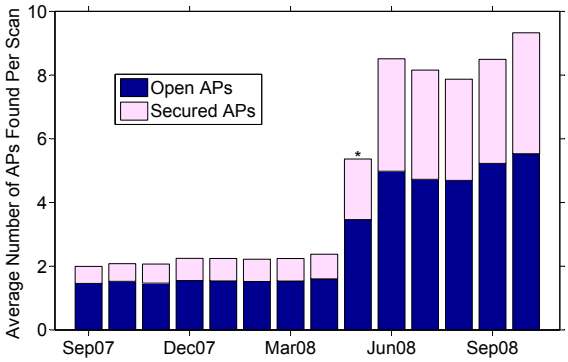


Figure 2.5. Background: The number of APs found per scan over a 13 month period.

tion and obtaining an IP address, the bus pinged a known server through the AP, to make sure the associated AP is open. Once verified, the bus uploaded/downloaded data to a lab server through the AP. We use several optimizations such as DHCP-caching to reduce the time to associate to an AP. Figure 2.5 shows the average number of APs found by a bus during a single scan over a 13 month period. On average, over 100 unique APs were encountered by the buses per day.

The bus-AP encounters form an intermittently connected network, where the bus is connected to the Internet for a few 10's of seconds through an AP before being disconnected for a few minutes [24]. In this thesis, I evaluate Thedu (Chapter 4) using the Dome-Infrastructure testbed.

Dome-Mesh

In 2007, the Dome testbed was diversified to include a mesh network. In cooperation with the Town of Amherst, a 26 node WiFi mesh testbed was installed in the Amherst town center. The testbed consists of lightweight access points managed by a central controller. All the mesh nodes advertise the same SSID to facilitate seamless hand-off. The APs use two radios: an 802.11g radio for the public and mobile nodes to connect to, and an 802.11a radio for AP interactions. The mesh network provides Internet connectivity to both passengers in the bus and stationary users in the town center.

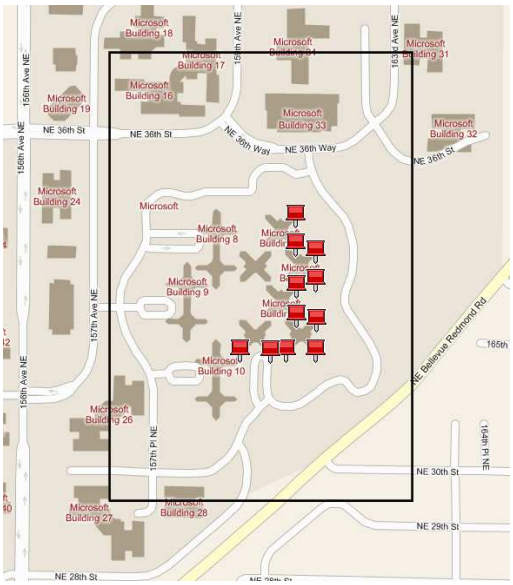


Figure 2.6. Background: The layout of APs in the VanLAN testbed. The thumbtacks represent the position of the APs and the black rectangle shows that region where vans can receive packets from the APs.

In this thesis, I evaluate ViFi (Chapter 5) using trace-driven simulations on the Dome-Mesh testbed.

Dome-3G

Finally, the GPRS modems on the buses were upgraded with Sierra Wireless 881 3G USB Modems in 2008. The 3G modem has HSDPA-based service via AT&T. The vehicles visit many locations multiple times each day. This set up allows us to analyze the stationarity of 3G availability with respect to location and time of day. I use the Dome-3G testbed to conduct measurement studies on 3G availability and to evaluate the Wiffler protocol (Chapter 6).

2.2.2 VanLAN

The second testbed used in this thesis is the VanLAN testbed, deployed in the Microsoft campus in Redmond. The VanLAN testbed is a vehicular-mesh testbed consisting of a 11-node mesh and two vehicles.

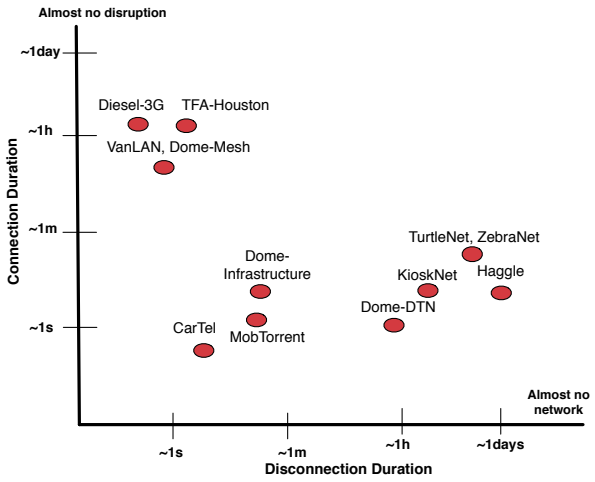


Figure 2.7. Background: Survey of mobile testbeds with respect to connection and disconnection durations.

The APs in VanLAN are deployed across five buildings in the Microsoft campus, as illustrated in Figure 2.6, covering a 828×559 sq meter area. The vehicles provide a shuttle service around the town, moving within a speed limit of about 40 Km/h.

The APs and vehicles have small desktops with Atheros 5213 chipset radios. The antennae are omnidirectional and are mounted on the roofs of the respective buildings and vehicles. All nodes are set to the same 802.11 channel for the experiments.

I use the VanLAN testbed to study different handoff policies in a vehicular-mesh environment, and to evaluate the ViFi (Chapter 5) protocol.

2.2.3 Survey of mobile testbeds

Several wireless testbeds have been deployed to study mobile connectivity. Figure 2.7 quantitatively presents the different testbeds by the reported observed median frequency of disconnection and connection. The figure also places the testbeds in the context of the Dome and VanLAN testbeds.

DTN testbeds: KioskNet [96] is a testbed deployed in developing regions to provide Internet access to rural areas. The Internet gateway is available in cities,

and the rural areas are equipped with several kiosks. Vehicles that move between the city and the villages transfer data between the kiosks and the Internet gateways. TurtleNet [100] and ZebraNet [117] testbeds are sparse networks deployed to monitor animals in their natural habitats. Haggle [71] is a DTN testbed deployed in Cambridge, England, consisting of bluetooth-enabled devices carried by humans. The devices communicate when within range of each other. In all of the DTN testbeds, connectivity between nodes were unpredictable because the nodes do not always move in a predefined pattern.

Testbeds with intermittent connectivity: The CarTel [61, 49] testbed comprises of 27 cabs operating in the Boston area. The cabs get intermittent Internet access when they are within range of open WiFi APs. The mean bus-AP contact observed in the CarTel testbed is 10 seconds. The MobTorrent [40] testbed deployed in Singapore is also an intermittently connected testbed, where 16 public transit buses get Internet connectivity using open WiFi APs. The mean bus-AP contact observed in the MobTorrent testbed is 15 seconds. In the intermittently connected testbeds, the interactions between the vehicle and the infrastructure were short-lived and often unpredictable.

Mesh testbeds: The TFA-Houston [53] testbed deployed in Rice university is a 4000-user urban mesh testbed. Vehicles in the testbed move in a 2.5 km loop and connect to the Internet via the mesh nodes. The authors studied the performance of different handoff policies in the vehicular mesh environment, similar to the VanLAN study that we conducted.

The above testbeds were deployed to conduct experiments in specific wireless environments such as DTNs or meshes. The Dome (including Dome-DTN, Dome-Infrastructure, Dome-Mesh and Dome-3G) and the VanLAN testbed together provide a diverse platform that enables measurements and evaluation in a wide range of mobile environments.

2.3 Opportunistic mechanisms

The underlying technique used to design protocols in this thesis is *opportunistic resource usage*. Below, I describe the four opportunistic mechanisms used in this work, and elaborate on the resource management challenge that they pose. I also discuss the rationale behind choosing the specific opportunistic mechanism for the network environment.

2.3.1 Packet replication in DTNs

Replication is an opportunistic mechanism involving routing multiple copies of the packet towards the destination. In traditional *forwarding-based* routing [90, 66], there is at most one copy of a packet in the system that is forwarded, at each step, to the optimal next-hop node. However, in DTNs it is difficult to determine the optimal next-hop because of the uncertainty in topology and limited knowledge about the environment. Therefore, a forwarding-only protocol is not suitable for the DTN environment where the single copy may be lost if the packet is forwarded to a dead-end node.

In contrast, *replication-based protocols* do not compute the optimal next-hop; instead, they opportunistically route the same copy of the packet to multiple nodes during contacts. Replication considerably improves packet delivery under uncertain topology conditions. However, naive replication wastes bandwidth and can severely degrade performance. There has been several related works that use replication for DTN routing and these works limit replication in various ways: *(i)* using historic meeting information [32, 31, 74]; *(ii)* removing useless packets using acknowledgments of delivered data [31]; *(iii)* using probabilistic mobility information to infer delivery [99]; *(iv)* using network coding [112] and coding with redundancy [62]; and *(v)* bounding the number of replicas of a packet [102, 99, 78].

However, the effect of the various replication decisions on the bandwidth resource and on the routing metric is unclear. In contrast, RAPID (Chapter 3) uses a utility-based replication protocol that tunes replication according to the specified metric and the available resources.

2.3.2 Prefetching in intermittently connected networks

The second opportunistic mechanism is aggressive prefetching. Prefetching involves opportunistically fetching application data when bandwidth becomes available, in anticipation that a user will request for the data. Clearly, prefetching can improve performance of interactive applications in networks with frequent disconnections, if the user request is already prefetched before the disruption occurs. However, aggressive prefetching poses a resource management problem: Given limited bandwidth opportunity, what application data to prefetch?

Prefetching has been used to mask disruptions in several scenarios: Chandra et al. [38] used prefetching to improve availability or response time for disconnected Web operation. Padmanabhan and Mogul [88] proposed a markov model for predicting and prefetching the most popular web requests. The TEK [94] system is an email-based web browser that compresses prefetched search results into an email. Jiang and Kleinrock [65] proposed a technique where a connected client prefetches files during idle time based on the probability that the user will request the file.

In this thesis, I present Thedu (Chapter 4), a protocol that uses prefetching to opportunistically fetch web pages in response to a web search request. To address the resource management challenge, Thedu uses a combination of IR techniques and utility-driven prioritization to ensure that the most useful web pages are prefetched during the short bandwidth opportunity.

2.3.3 Opportunistic forwarding for handoffs in WiFi meshes

The third opportunistic mechanism is opportunistic forwarding. This mechanism leverages opportunistic overhearing. When a packet is sent in the wireless medium, several nearby APs can overhear the packet. Leveraging packet overhearing can help retrieve dropped packets. The challenge is in coordinating among the APs to retrieve a packet, without knowing which subset of APs overheard the packet.

Leveraging overheard packets is inspired by cellular networks [111], where multiple APs act in concert to improve client performance. The cellular methods, however, require tight integration with the physical layer and strict timing across APs. These abilities are not available in WiFi networks.

Opportunistic routing protocols such as ExOR [27] and MORE [36] leverage overhearing to improve throughput in static mesh networks. Their approach requires packet batching to leverage overhearing; the authors recommend using a batch size of at least around ten. Although the batching approach improves throughput, it cannot be used to support interactive applications, because of the delays associated with batching packets.

In this thesis, I present ViFi (Chapter 5), a protocol that exploits opportunistic overhearing to enable interactive applications in vehicular-mesh environments. ViFi uses utility-driven sender prioritization protocol to assign relaying probabilities to each node. Every node opportunistically forwards overheard packets according to its relaying probability, such that collectively the relaying decision is effective. The protocol is decentralized, operates on individual packets (as opposed to batching), and is well-suited for interactive applications.

2.3.4 Opportunistic augmentation to conserve 3G spectrum

The fourth opportunistic mechanism is opportunistic augmentation. Specifically, my focus is on augmenting 3G networks using opportunistic WiFi access for vehicular

users. Cellular users, especially in big cities, are facing poor performance because the 3G spectrum is severely overloaded [115]. The key idea is to reduce the load on 3G by using WiFi connectivity when possible. Augmenting 3G using WiFi access is challenging for vehicular users. WiFi APs have a short range; even when APs are in range, the quality of connectivity may be poor [61, 58]. Thus, it is unclear if WiFi can usefully augment 3G, while providing the ubiquity and reliability that 3G subscribers expect.

Many previous works propose mobile systems that augment one network with another. One method is to select the interface with low idle power consumption to wake up another interface, to save energy [98, 26]. Zhong et al. [91] estimate the power consumption of different interfaces for various network activities. They use these estimates to switch between the different interfaces to save energy. Vertical handoff techniques select the interface that currently offers the best performance [30]. Striping techniques multiplex data across different interfaces to balance load and improve performance [93].

In this thesis, I present Wiffler (Chapter 6), a protocol that augments 3G with opportunistic WiFi connectivity. In contrast to related works, the primary goal is not to optimize power consumption or performance. Instead, Wiffler uses a cheaper but unreliable WiFi network to augment a more expensive but reliable 3G network. Wiffler uses WiFi availability prediction to maximize data offload while satisfying application requirements.

2.4 Lessons learnt in applying opportunistic techniques

Based on my experience in applying opportunistic techniques to network problems, I present some intuitions for the question – what network conditions are necessary for a given opportunistic mechanism to be useful?

Replication: Opportunistic replication of packets is most useful in multi-hop networks that exhibit high uncertainty. In a separate work (not part of this thesis), we show that the replication gain is high when there is high variance in path delays [109], where the replication gain is the delay when using replication divided by the delay when not using replication. In DTNs, path delays have high variance because of node mobility and unpredictable network topology. As a result, opportunistic replication improves delay performance in DTNs. We show that opportunistic replication can improve application performance in multi-hop intermittently connected networks as well (Chapter 4). However, the delay variance in well-connected networks is typically low, and we show that replication is unlikely to provide performance benefits [109].

Prefetching: The usefulness of prefetching depends on the characteristics of the application. For example, for applications such as web search or web browsing, future user requests can be predicted based on current request [88]. For these applications, prefetching data can improve application latency, especially in intermittent connected environments. However, for bulk transfer applications or VoIP applications, prefetching is unlikely to be useful because in both cases, future data requests cannot be predicted based on the current data.

Opportunistic forwarding: Opportunistic forwarding can be exploited when the channel allows overhearing by neighbor nodes and when the probability of packet reception is non-deterministic and independent across nodes. When packet reception is independent, there is high probability that even if the intended next hop does not receive the packet, other nodes in the vicinity will overhear the packet. In ViFi, we opportunistically forward overheard packets to reduce disruptions caused by packet losses. Opportunistic forwarding has also been exploited in multi-hop mesh networks [27, 36], where the farthest node that overhears a packet opportunistically forwards the packet, reducing packet delay and improving throughput. In disconnected networks, exploiting overhearing is less useful because opportunistic forwarding only

reduces transmission delays and not the queuing delay caused by large inter-contact times. In disconnected networks, the transmission delay is insignificant compared to the queuing delays.

Opportunistic augmentation: Opportunistic augmentation is possible when users can exploit diversity in terms of network, channel, or technology.There are various scenarios when such diversity is available. For example, when multiple providers provide Internet access to a device, a user can exploit the network diversity across the different providers to augment connectivity and improve application performance [93]. In Wiffler, we focus on devices that can connect over both 3G and WiFi, and exploit the technology diversity to augment 3G with WiFi.

CHAPTER 3

RAPID: ROUTING IN DISCONNECTED NETWORKS

DTNs have the potential to enable connectivity in a vast number of disconnected scenarios where it is difficult or expensive to deploy infrastructure. For example, the potential of DTNs can be harnessed by projects such as TIER [6], Digital Study Hall [51], and One Laptop Per Child [2] to extend the reach of the Internet to rural and developing regions. However, applications that work well in wired and wireless networks simply breakdown in the DTN environment because of the lack of a contemporaneous end-to-end path. In fact, the lack of end-to-end path, frequent topology changes, and extremely limited information about the network, make DTNs one of the most challenging environments for communication.

In this chapter, I describe RAPID, a DTN routing protocol that overcomes these challenges to reliably route data. RAPID is a replication based routing protocol that replicating multiple copies of the packet during *opportunistic* contacts with other nodes. First, I present a measurement study to show that existing routing protocols designed for wireless networks perform poorly in the DTN environment, and replication can significantly improve performance. However, the measurement study also shows that naive replication hurts performance when resources are constrained.

The goal of RAPID is to address the resource management challenge by answering the following question: *During a limited bandwidth connection between two nodes, what subset of packets should be replicated?*. Clearly, all packets in the buffer cannot be replicated because the contact is of limited duration. However, replicating the same packet over and over again improves the performance of that packet at the cost of other

packets in the buffer.The key insight in RAPID is to formulate the routing problem as a *resource allocation problem*. RAPID carefully allocates the limited bandwidth resource to a subset of packets to reduce resource wastage and improve routing performance; in other words, RAPID prioritizes packets according to the available resources and the routing metric and replicates the packets in the prioritized order.

To this end, RAPID uses a utility-driven protocol that tunes replication according to the given routing metric. RAPID translates the metric to a per-packet utility function. During an opportunistic contact, RAPID determines if the marginal utility of replicating a packet justifies the resources used. RAPID then replicates packets in the decreasing order of the marginal utility of replication. We instantiate RAPID to optimize three different metrics—average delay, worse-case delay and delivery within a deadline.

RAPID is a heuristic approach, but we prove two hardness results to substantiate the need for a heuristic. We prove that: (1) online DTN routing algorithms without complete future knowledge can perform arbitrarily far from optimal, and, (2) designing an offline DTN routing algorithm with complete future knowledge is computationally hard. Both of the results are with respect to the delay metric.

The rest of this chapter is organized as follows—In Section 3.1, I describe the state of the art in DTN routing. In Section 3.2, I present a measurement study to show that packet replication can significantly improve routing performance in DTNs, but can also waste resources. In Sections 3.3 and 3.4, I describe the RAPID protocol and implementation details, respectively. In Section 3.5, I present the case for the heuristic approach that RAPID takes. In Section 3.6, I describe the evaluation of RAPID and Section 3.7 concludes this chapter.

3.1 Related work

Forwarding vs Replication We classify existing DTN routing protocols as those that replicate packets and those that forward only a single copy.

Forwarding routing protocols maintain at most one copy of a packet in the network [62, 67, 107]. Jain et al. [62] propose a forwarding protocol to minimize the average delay of packet delivery using oracles with varying degrees of future knowledge. Our deployment experience suggests that, even for a scheduled bus service, implementing the simplest oracle is difficult; connection opportunities are affected by many factors in practice including weather, radio interference, and system failure. Jones et al. [67] propose a link-state protocol based on epidemic propagation to disseminate global knowledge, but use a single path to forward a packet. Shah et al. [97] and Spyropoulos et al. [107] present an analytical framework for the forwarding-only case assuming a grid-based mobility model.

The consensus [102] is that replicating packets can improve performance over just forwarding, but risk degrading performance when resources are limited.

Replication routing protocols replicate multiple copies of the packets at transfer opportunities hoping to find a path to a destination. However, naive replication wastes resources and can severely degrade performance. Proposed protocols use several heuristics to control replication, but the effect of a heuristics on the routing metric is unclear, as described in Section 2.3.1. In contrast, RAPID explicitly calculates the effect of replication on the given routing metric while accounting for resource constraints.

Resource Constraints RAPID also differs from most previous work in its assumptions regarding resource constraints, routing policy, and mobility patterns. Table 3.1 shows a taxonomy of many existing DTN routing protocols based on assumptions about *bandwidth* available during transfer opportunities and the *storage* carried by nodes; both are either *finite* or *unlimited*. For each work, we state in

	Storage	Bandwidth	Routing	Previous work (and mobility)
P1	Unlimited	Unlimited	Replication	Epidemic [78], Spray and Wait [102]: Constraint in the form of channel contention (Grid-based synthetic)
P2	Unlimited	Unlimited	Forwarding	Modified Djikstra's algorithm Jain et al. [62] (simple graph), MobySpace [70] (Powerlaw)
P3	Finite	Unlimited	Replication	SWIM [99] (Exponential), MV [32] (Community-based synthetic), Prophet [74] (Community-based synthetic)
P4	Finite	Finite	Forwarding	Jones et al. [67] (AP traces), Jain et al. [62] (Synthetic DTN topology)
P5	Finite	Finite	Replication	RAPID (Vehicular DTN traces, testbed deployment), MaxProp [31] (Vehicular DTN traces)

Table 3.1. RAPID: A classification of related work into DTN routing scenarios.

parentheses the mobility model used. RAPID is a replication-based protocol that assumes constraints on both storage and bandwidth (P5) — the most challenging and most practical problem space.

P1 and P2 are important to examine for valuable insights that theoretical tractability yields but are impractical for real DTNs with limited resources. Many studies [74, 32, 99] analyze the case where storage at nodes is limited, but bandwidth is unlimited (P3). This scenario may happen when the radios used and the duration of contacts allow transmission of more data than can be stored by the node. However, we find this scenario to be uncommon — typically storage is inexpensive and energy efficient. For mobile DTNs, and especially vehicular DTNs, transfer opportunities are short-lived [61, 31].

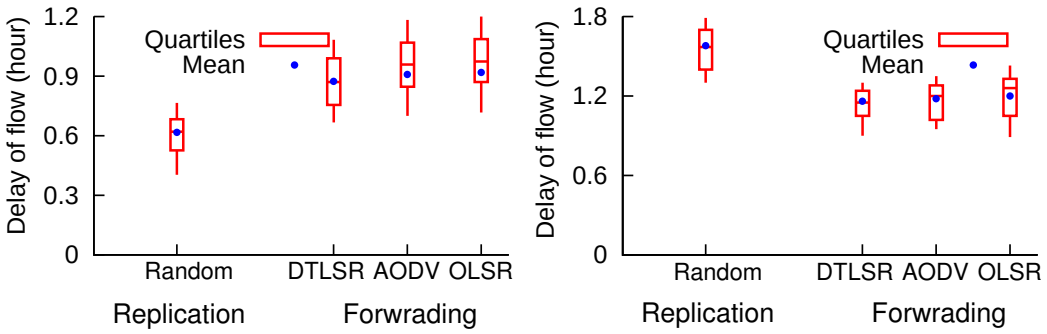
Some theoretical works [118, 103, 99] derive closed-form expressions for average delay and number of replicas in the system as a function of the number of nodes and mobility patterns. Although these analyses contributed to important insights in the design of RAPID, their assumptions about mobility patterns or unlimited resources are too restrictive to be applicable in practice.

3.2 Measurement

State-of-the-art wireless routing protocols such as AODV [90] and OLSR [3] are designed for well-connected networks where there is end-to-end connectivity between nodes. Such routing protocols use a *forwarding* only strategy and do not replicate packets through multiple paths. Some DTN routing protocols such as DTLSR [45] also use a forwarding only strategy to route packets. However, replication is known to improve performance in sparse DTNs. To understand the performance of replication-based protocols and forwarding-based protocols, we conduct a simple experiment comparing the performance of *Random*, AODV [90], OLSR [3], and DTLSR [45]. Random is a naive replication-based routing protocol that randomly replicates packets during a transfer opportunity.

The evaluation is based on traces collected from Dome-DTN (described in Section 2.2) over a 3 day period. To adapt AODV and OLSR to DTNs, we set a high timeout value allowing them to buffer packets. Each experiment involves 30 concurrent flows corresponding to 30 randomly chosen node pairs. We perform trace-driven simulations using QualNet [4] with a moderate load of 20pkt/flow/hour and a high load of 50pkt/flow/hour.

Figure 3.1(a) shows that in Dome-DTN environment, random replication yields 1.5× reduction in delay compared to traditional, forwarding-based, routing protocols. Instead, even randomly replicating packets, the most naive replication-based protocol can improve delay performance. Figure 3.1(b) shows that under high load, replication increases the delay by 15% over forwarding. Replication hurts performance when resources are limited, as is the case when the offered load is high. This measurement study is published as part of a technical report [109].



(a) Dome-DTN: Moderate load (b) Dome-DTN: High load

Figure 3.1. RAPID: Each boxplot shows min, max, 25%, 75% quartiles, median, and mean packet delays. Replication benefits significantly in Dome-DTN but can hurt performance under high load.

3.3 Protocol description

RAPID models DTN routing as a utility-driven resource allocation problem. The model is inspired by the seminal work by Kelly et al. [68], who show that network optimization problems can be formulated as a resource allocation problem. The authors solve the allocation problem using network utility maximization, where the utilities are defined as a function of the optimization variables. The utility function is typically a concave function and in several instances, the allocation problem is solved in a decentralized manner using feedback from the network [68]. The RAPID protocol is similar in spirit to Kelly's framework. However, in DTNs, the utility function may not always be concave, feedback is limited, and the utility function is difficult to model because of node mobility. Below, we describe how RAPID solves the resource allocation problem for the DTN environment.

In RAPID, a packet is routed by replicating it until a copy reaches the destination. The key question is: given limited bandwidth, how should packets be replicated in the network so as to optimize a specified *routing metric*? RAPID derives a per-packet *utility function* from the routing metric. At a transfer opportunity, it replicates a packet that locally results in the largest increase in utility.

Consider a routing metric such as *average delay of packets*. The corresponding utility U_i of packet i is the negative of the expected delay to deliver i, i.e., the time i has already spent in the system plus the additional expected delay before i is delivered. Let δU_i denote the increase in U_i by replicating i and s_i denote the size of i. Then, RAPID replicates the packet with the largest value of $\delta U_i/s_i$ among packets in its buffer; in other words, the packet with the largest marginal utility.

In general, U_i is defined as the expected contribution of i to the given routing metric. For example, the metric *minimize average delay* is measured by summing the delay of packets. Accordingly, the utility of a packet is its expected delay. Thus, RAPID is a heuristic based on locally optimizing marginal utility, i.e., the expected increase in utility per unit resource used. RAPID replicates packets in decreasing order of their marginal utility at each transfer opportunity.

The marginal utility heuristic has some desirable properties. The marginal utility of replicating a packet to a node is low when *(i)* the packet has many replicas, or *(ii)* the node is a poor choice with respect to the routing metric, or *(iii)* the resources used do not justify the benefit. For example, if nodes meet each other uniformly, then a packet i with 6 replicas has lower marginal utility of replication compared to a packet j with just 2 replicas. On the other hand, if the peer is unlikely to meet j's destination for a long time, then i may take priority over j.

3.3.1 System Model

We model a DTN as a set of mobile nodes. Two nodes transfer data packets to each other when within communication range. During a transfer, the sender replicates packets while retaining a copy. A node can deliver packets to a destination node directly or via intermediate nodes, but packets may not be fragmented. There is limited storage and transfer bandwidth available to nodes. Destination nodes are

assumed to have sufficient capacity to store delivered packets, so only storage for in-transit data is limited. Node meetings are assumed to be short-lived.

Formally, a DTN consists of a node meeting schedule and a workload. The node meeting schedule is a directed multigraph $G = (V, E)$, where V and E represent the set of nodes and edges, respectively. Each directed edge e between two nodes represents a meeting between them, and it is annotated with a tuple (t_e, s_e), where t is the time of the meeting and s is the size of the transfer opportunity. The workload is a set of packets $P = \{(u_1, v_1, s_1, t_1), (u_2, v_2, s_2, t_2), \ldots\}$, where the ith tuple represents the source, destination, size, and time of creation (at the source), respectively, of packet i. The goal of a DTN routing algorithm is to deliver all packets using a feasible schedule of packet transfers, where *feasible* means that the total size of packets transfered during each opportunity is less than the size of the opportunity, always respecting storage constraints.

In comparison to Jain et al.[62] who model link properties as continuous functions of time, our model assumes discrete short-lived transfers; this makes the problem analytically more tractable and characterizes many practical DTNs well.

3.3.2 RAPID overview

Figure 3.2 shows an overview of RAPID. The RAPID protocol is executed when two nodes X and Y come within communication range, and the protocol is symmetric. RAPID has two components, the Inference algorithm and the Selection algorithm. In the inference algorithm, X estimates the marginal utility of replicating each packet i in its buffer to Y, and vice versa. The Selection algorithm then replicates packets in the decreasing order of marginal utility.

3.3.3 Selection algorithm

The RAPID protocol executes when two nodes are within radio range and have discovered one another. The protocol is symmetric; without loss of generality, and

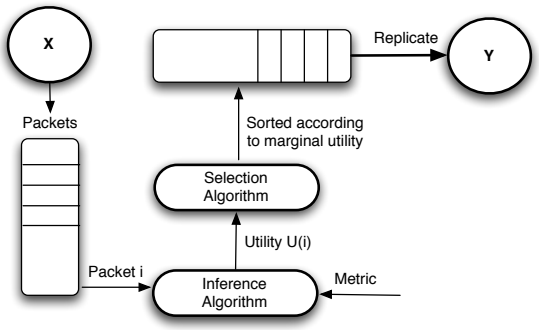


Figure 3.2. RAPID: Protocol overview.

$D(i)$	Packet i's expected delay = $T(i) + A(i)$
$T(i)$	Time since creation of i
$a(i)$	Random variable that determines the remaining time to deliver i
$A(i)$	Expected remaining time = E$[a(i)]$

Table 3.2. RAPID: List of commonly used variables.

describes how node X determines which packets to transfer to node Y (refer to the box marked PROTOCOL RAPID).

RAPID also adapts to storage restrictions for in-transit data. If a node exhausts all available storage, packets with the lowest utility are deleted first as they contribute least to overall performance. However, a source never deletes its own packet unless it receives an acknowledgment for the packet.

3.3.4 Inference algorithm

Next, we describe how PROTOCOL RAPID can support specific metrics using an algorithm to infer utilities. Table 3.2 defines the relevant variables.

PROTOCOL RAPID(X, Y):

1. *Initialization*: Obtain metadata from Y about packets in its buffer and metadata Y collected over past meetings (detailed in Section 3.4.2).
2. *Direct delivery*: Deliver packets destined to Y in decreasing order of their utility.
3. *Replication*: For each packet i in node X's buffer
   (a) If i is already in Y's buffer (as determined from the metadata), ignore i.
   (b) Estimate marginal utility, δU_i, of replicating i to Y.
   (c) Replicate packets in decreasing order of $\frac{\delta U_i}{s_i}$.
4. *Termination*: End transfer when out of radio range or all packets replicated.

3.3.4.1 Metric 1: Average delay

To minimize the average delay of packets in the network we define the utility of a packet as

$$U_i = -D(i) \tag{3.1}$$

since the packet's expected delay is its contribution to the performance metric. Thus, the protocol attempts to greedily replicate the packet whose replication reduces the delay by the most among all packets in its buffer.

3.3.4.2 Metric 2: Missed deadlines

To minimize the number of packets that miss their deadlines, the utility is defined as the probability that the packet will be delivered within its deadline:

$$U_i = \begin{cases} P(a(i) < L(i) - T(i)), & L(i) > T(i) \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

where $L(i)$ is the packet life-time which is an input parameter. A packet that has missed its deadline can no longer improve performance and is thus assigned a value

of 0. The marginal utility is the improvement in the probability that the packet will be delivered within its deadline, so the protocol replicates the packet that yields the largest improvement among packets in its buffer.

3.3.4.3 Metric 3: Worse-case delay

To minimize the worse-case delay of packets in the network, we define the utility U_i as

$$U_i = \begin{cases} -D(i), & D(i) \geq D(j) \quad \forall j \in S \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

where S denotes the set of all packets in X's buffer. Thus, U_i is the negative expected delay if i is a packet with the maximum expected delay among all packets held by Y. So, replication is useful only for the packet whose expected delay is largest. For the routing algorithm to be work conserving, RAPID computes the utility for the packet whose delay is currently the largest; i.e., once a packet with largest delay is evaluated for replication, the utility of the remaining packets is recalculated using Eq. 3.3.

3.4 Estimating utilities

How does a RAPID node estimate expected delay in Eqs. 3.1 and 3.3, or the probability of packet delivery within a deadline in Eq. 3.2? The expected delivery delay is the minimum expected time until any node with the replica of the packet delivers the packet; so a node needs to know which other nodes possess replicas of the packet and when they expect to meet the destination.

To estimate expected delay we assume that the packet is delivered directly to the destination, ignoring the effect of further replication. This estimation is nontrivial even with an accurate global snapshot of system state. For ease of exposition, we first

ALGORITHM ESTIMATE_DELAY(X, Q, Z):
Node X with a set of packets Q to destination Z estimates the time, $A(i)$, until packet $i \in Q$ is delivered to Z as follows:

1. Sort packets in Q in decreasing order of $T(i)$. Let $b(i)$ be the sum of sizes of packets that precede i, and B the expected transfer opportunity in bytes between X and Z (refer Figure 3.3).
2. X by itself requires $\lceil b(i)/B \rceil$ meetings with Z to deliver i. Let M_{XZ} be the inter contact time between X and Z. Compute the random variable $M_X(i)$, the delay of packet i if only X where delivering the packet, as
$$M_X(i) = M_{XZ} + M_{XZ} + \ldots \lceil b(i)/B \rceil \text{ times} \tag{3.4}$$
3. Let $X_1, \ldots, X_k$ be the set of nodes possessing a replica of i. Estimate remaining time $a(i)$ as
$$\mathrm{a}(i) = \min(M_{X_1}(i), \ldots, M_{X_k}(i)) \tag{3.5}$$
4. Expected delay $D(i) = T(i) + E[a(i)]$

present RAPID's estimation algorithm as if we had knowledge of the global system state, and then we present a practical distributed implementation.

3.4.1 Algorithm Estimate_Delay

A RAPID node uses the algorithm ESTIMATE_DELAY to estimate the delay of a packet in its buffer. ESTIMATE_DELAY works as follows (refer to box marked

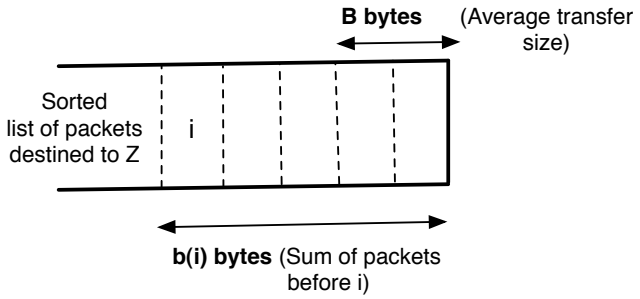


Figure 3.3. RAPID: Position of packet i in a queue of packets destined to Z.

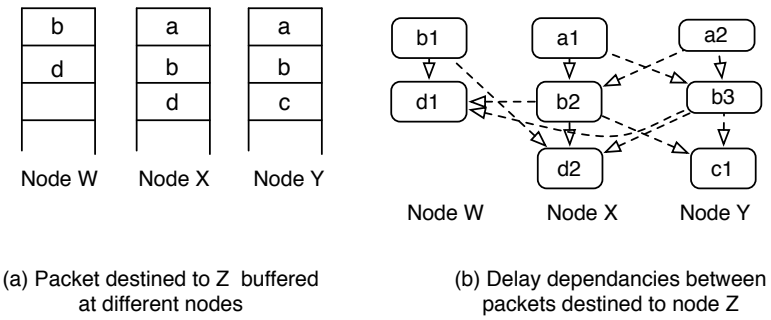


Figure 3.4. RAPID: Delay dependencies between packets destined to node Z buffered in different nodes.

ALGORITHM ESTIMATE_DELAY): In Step 1, each node X maintains a separate queue of packets, Q, destined for each node Z sorted in decreasing order of creation times; this is the order in which the packets will be delivered when X meets Z in PROTOCOL RAPID. In Step 2 of ESTIMATE_DELAY, X computes the delivery delay distribution of packet i if delivered directly by X. In Step 3, X computes the minimum across all replicas of the corresponding delivery delay distributions; we note that the delivery time of i is the time until the first node delivers the packet. ESTIMATE_DELAY assumes that the meeting time distribution is the same as the inter-meeting time distribution.

ESTIMATE_DELAY makes a simplifying independence assumption that does not hold in general. Consider Figure 3.4(a), an example showing the positions of packet replicas in the queues of different nodes. All packets have a common destination Z and each queue is sorted by $T(i)$. Assume that the transfer opportunities and packets are of unit-size.

In Figure 3.4(a), packet b may be delivered in two ways: *(i)* if W meets Z; *(ii)* one of X and Y meets Z and then one of X and Y meet Z again. These delay dependencies can be represented using a dependency graph as illustrated in Fig 3.4(b); packets with the same letter and different indices are replicas. A vertex corresponds to a packet

replica. An edge from one node to another indicates a dependency between the delays of the corresponding packets. Recall that M_{XY} is the random variable that represents the meeting time between X and Y.

ESTIMATE_DELAY ignores all the non-vertical dependencies. For example, it estimates b's delivery time distribution as

$$\min(M_{WZ}, M_{XZ} + M_{XZ},\ M_{YZ} + M_{YZ}),$$

whereas the distribution is actually

$$\min(M_{WZ}, \min(M_{XZ}, M_{YZ}) + \min(M_{XZ}, M_{YZ})).$$

Estimating delays without ignoring the non-vertical dependancies is challenging. Using a simplifying assumption that the transfer opportunities and packets are unit-sized, we design algorithm DAG_DELAY (described in Appendix A.2, that estimates the expected delay by taking into account non-vertical dependancies. Although DAG_DELAY is of theoretical interest, it cannot be implemented in practice because DAG_DELAY assumes that — *(i)* the transfer opportunity size is exactly equal to the size of a packet.This assumption is fundamental for the design of DAG_DELAY, and *(ii)* nodes have a global view of the system.

In general, ignoring non-vertical edges can arbitrarily inflate delay estimates for some pathological cases (detailed in Appendix A.2). However, we find that using ESTIMATE_DELAY makes our implementation *(i) simple* — computing an accurate estimate is much more complex especially when transfer opportunities are not unit-sized as above — and *(ii) distributed* — in practice, RAPID does not have global view, but ESTIMATE_DELAY can be implemented using a thin in-band control channel, as we describe in Section 3.4.2.

3.4.1.1 Exponential distributions

We walk through the distributed implementation of ESTIMATE_DELAY for a scenario where the inter-meeting time between nodes is exponentially distributed. Further, suppose all nodes meet according to a uniform exponential distribution with mean time $1/\lambda$. In the absence of bandwidth restrictions, the expected delivery delay when there are k replicas is the mean meeting time divided by k, i.e., $\mathrm{P}(a(i) < t) = 1 - e^{-k\lambda t}$ and $A(i) = \frac{1}{k\lambda}$. (Note that the minimum of k i.i.d. exponentials is also an exponential with mean $1/k$ of the mean of the i.i.d exponentials [35].)

However, when transfer opportunities are limited, the expected delay depends on the packet's position in nodes' buffers. In Step 2 of ESTIMATE_DELAY, the time for some node X to meet the destination $\lceil b(i)/B \rceil$ times is described by a $\lceil b(i)/B \rceil$ order Erlang distribution with mean $\frac{1}{\lambda} \cdot \lceil b(i)/B \rceil$.

If packet i is replicated at k nodes, Step 3 computes the delay distribution $a(i)$ as the minimum of k gamma variables. We do not know of a closed form expression for the minimum of gamma variables. Instead, if we assume that the time taken for a node to meet the destination $b(i)/B$ times is exponential with the same mean $\frac{1}{\lambda} \cdot \lceil b(i)/B \rceil$, we can again estimate $a(i)$ as the minimum of k exponentials as follows.

Let $n_1(i), n_2(i), \ldots, n_k(i)$ be the number of times each of the k nodes respectively needs to meet the destination to deliver i directly. Then $A(i)$ is computed as:

$$\mathrm{P}(a(i) < t) = 1 - e^{-(\frac{\lambda}{n_1(i)} + \frac{\lambda}{n_2(i)} + \ldots + \frac{\lambda}{n_k(i)})t} \tag{3.6}$$

$$A(i) = \frac{1}{\frac{\lambda}{n_1(i)} + \frac{\lambda}{n_2(i)} + \ldots + \frac{\lambda}{n_k(i)}} \tag{3.7}$$

When the meeting time distributions between nodes are non-uniform, say with means $1/\lambda_1, 1/\lambda_2 \ldots 1/\lambda_k$ respectively, then $A(i) = (\frac{\lambda_1}{n_1(i)} + \frac{\lambda_2}{n_2(i)} + \ldots + \frac{\lambda_k}{n_k(i)})^{-1}$.

3.4.1.2 Unknown mobility distributions

To estimate mean inter-node meeting times in the Dome-DTN testbed, every node tabulates the average time to meet every other node based on past meeting times. Nodes exchange this table as part of metadata exchanges (Step 1 in PROTOCOL RAPID). A node combines the metadata into a meeting-time adjacency matrix and the information is updated after each transfer opportunity. The matrix contains the expected time for two nodes to meet directly, calculated as the average of past meetings.

Node X estimates $E(M_{XZ})$, the expected time to meet Z, using the meeting-time matrix. $E(M_{XZ})$ is estimated as the expected time taken for X to meet Z in at most h hops. (Unlike uniform exponential mobility models, some nodes in the trace never meet directly.) For example, if X meets Z via an intermediary Y, the expected meeting time is the expected time for X to meet Y and then Y to meet Z in 2 hops. In our implementation we restrict $h = 3$. When two nodes never meet, even via three intermediate nodes, we set the expected inter-meeting time to infinity. Several DTN routing protocols [31, 74, 32] use similar techniques to estimate meeting probability among peers.

Let replicas of packet i destined to Z reside at nodes $X_1, \ldots, X_k$. Since we do not know the meeting time distributions, we simply assume they are exponentially distributed. Then from Eq. 3.7, the expected delay to deliver i is

$$A(i) = [\sum_{j=1}^{k} \frac{1}{E(M_{X_j Z}) \cdot n_j(i)}]^{-1} \tag{3.8}$$

We use an exponential distribution because bus meeting times in the testbed are very difficult to model. Buses change routes several times in one day, the inter-bus meeting distribution is noisy, and we found them hard to model even using mixture models. Approximating meeting times as exponentially distributed makes delay estimates easy to compute and performs well in practice.

3.4.2 Control channel

RAPID nodes needs to gather several state information to estimate utilities—number of replicas, positions of the replicas in each queue, the delivery delay of each queue etc. Previous studies [62] have shown that as nodes have the benefit of more information about the global system state, they can make significantly better routing decisions. We extend this idea to practical DTNs where no oracle is available. To this end, RAPID nodes gather knowledge about the global system state by disseminating metadata during the transfer opportunity.

RAPID uses an in-band *control channel* to exchange acknowledgments for delivered packets as well as metadata about every packet learnt from past exchanges. For each encountered packet i, RAPID maintains a list of nodes that carry the replica of i, and for each replica, an estimated time for delivery. Metadata for delivered packets is deleted when an ack is received.

A RAPID node sends the following information on encountering a peer.

- Average size of past transfer opportunities;
- Expected meeting times with nodes;
- List of packets delivered since last exchange;
- For each of its own packets, the updated delivery delay estimate based on current buffer state;
- Information about other packets if modified since last exchange with the peer.

For efficiency, a RAPID node maintains the time of last metadata exchange with its peers. The node only sends information about packets whose information changed since the last exchange, considerably reducing the size of exchange. When using the control channel, nodes have only an imperfect view of the system. The propagated information may be stale due to change is number of replicas, changes in delivery

delays, or if the packet is delivered but acknowledgments have not propagated. Nevertheless, our experiments confirm that *(i)* this inaccurate information is sufficient for RAPID to achieve significant performance gains over existing protocols and *(ii)* the overhead of metadata itself is small.

3.5 The case for a heuristic approach

RAPID is a heuristic protocol and there are two fundamental reasons for a heuristic approach. First, the inherent uncertainty of DTN environments rules out provably efficient online routing algorithms. Second, computing optimal solutions is hard even with complete knowledge about the environment. Both hardness results are formalized below

Theorem 1. *Let ALG be a deterministic online DTN routing algorithm with unlimited computational power.*

- *(a) If ALG has complete knowledge of a workload of n packets, but not of the schedule of node meetings, then it is $\Omega(n)$-competitive with an offline adversary with respect to the fraction of packets delivered.*
- *(b) If ALG has complete knowledge of the meeting schedule, but not of the packet workload, then it can deliver at most a third of packets compared to an optimal offline adversary.*

Theorem 2. *Given complete knowledge of node meetings and the packet workload* a priori*, computing a routing schedule that is optimal with respect to the number of packets delivered is NP-hard with an $\Omega(\sqrt{(}n))$ lower bound on approximability.*

The proofs are outlined in Appendix A.1 and formal proofs are presented in a technical report [18]. The hardness results naturally extend to the average delay metric for both the online as well as computationally limited algorithms.

Finally, traditional optimization frameworks for routing [50] and congestion control [68] based on fluid models appear difficult to extend to DTNs due to the inherently high feedback delay, uncertainty about network conditions, and the discrete nature of transfer opportunities that are more suited for transferring large "bundles" rather than small packets. Similarly, backpressure routing [11] is known to be optimal, in terms of throughput, for multihop wireless networks even without complete knowledge of the network. However, the backpressure result only holds for asymptotic cases. For fixed time intervals, designing an optimal routing protocol is not possible because of the hardness results with respect to delay, presented in Theorems 1 and 2.

3.6 Evaluation

3.6.1 Deployment

We implemented and deployed RAPID on our vehicular DTN testbed, DomeDTN [7] described in Section 2.2.

Each bus generates packets of size 1 KB according to a Poisson process. The destinations of the packets included only buses that were scheduled to be on the road, which avoided creation of many packets that could never be delivered. We did not provide the buses information about the location or route of other buses on the road. We set the default packet generation rate to 4 packets per hour generated for each pair of buses on the road.

Each bus executed RAPID to determine what packets to replicate during the transfer opportunity. When two buses are within communication distance, they exchanged the packets until the radios are out of range. During the experiments, the buses logged packet generation, packet delivery, delivery delay, meta-data size, and the total size of the transfer opportunity. Buses transfered random data after all routing was complete in order to measure the capacity and duration of each transfer opportunity. The

Avg. buses scheduled per day	19
Avg. total bytes transfered per day	261.4 MB
Avg. number of meetings per day	147.5
Percentage delivered per day	88%
Avg. packet delivery delay	91.7 min
Meta-data size/ bandwidth	0.002
Meta-data size/ data size	0.017

Table 3.3. Rapid: Average daily statistics of Rapid deployment

logs were periodically uploaded to a central server using open Internet APs found on the road.

3.6.1.1 Performance of deployed RAPID

We measured the routing performance of RAPID on the buses from Feb 6, 2007 until May 14, 2007[1]. The measurements are tabulated in Table 3.3. We exclude holidays and weekends since almost no buses were on the road, leaving 58 days of experiments. RAPID delivered 88% of packets with an average delivery delay of about 91 minutes. We also note that overhead due to meta-data accounts for less than 0.02% of the total available bandwidth and less than 1.7% of the data transmitted.

3.6.1.2 Validating the trace driven simulator

In the next section, we evaluate RAPID using a trace-driven simulator to stress test for varying network parameters. The simulator takes as input a schedule of node meetings, the bandwidth available at each meeting, and a routing algorithm. We validated our simulator by comparing simulation results against the 58-days of measurements from the deployment. In the simulator, we generate packets under the same assumptions as the deployment, using the same parameters for exponentially distributed inter-arrival times.

[1]The traces are available at `http://traces.cs.umass.edu`.

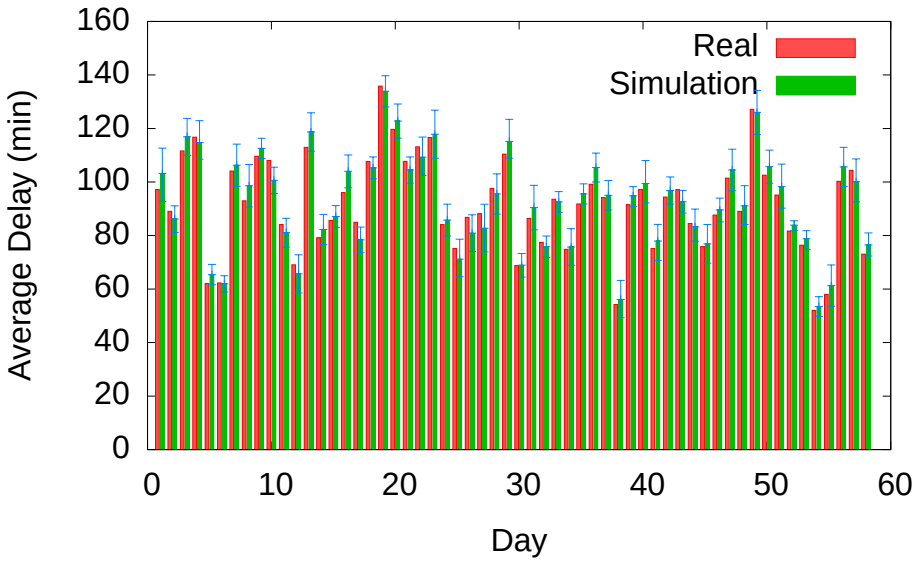


Figure 3.5. RAPID: Comparing deployment and simulation results over 58 days.

Figure 3.5 shows the average delay characteristics of the real system and the simulator. Delays measured using the simulator were averaged over the 30 runs and the error-bars show 95% confidence intervals. We find with 95% confidence that the simulator results are within 1% of the implementation measurement of average delay. The close correlation between system measurement and simulation increases our confidence in the accuracy of the simulator.

3.6.2 Trace-driven simulations

The goal of our trace-driven simulations is to show that, unlike existing work, RAPID can improve performance for customizable metrics. We evaluate RAPID using three metrics: worse-case, average delay, and missed deadlines. In all cases, we found that RAPID significantly outperforms existing protocols and also performs close to optimal for our workloads.

3.6.2.1 Experimental setup

Our evaluations are based on a custom trace-driven simulator described in Section 3.6.1.2. The meeting times between buses in these experiments are not known *a priori*. All values used by RAPID, including average meeting times, are learned during the experiment.

We compare RAPID to five other routing protocols: *MaxProp* [31], *Spray and Wait* [102], Prophet [74], *Random*, and *Optimal*. In all experiments, we include the cost of RAPID's in-band control channel for exchanging metadata.

MaxProp operates in a storage- and bandwidth-constrained environment, allows packet replication, and leverages delivery notifications to purge old replicas; of recent related work, it is closest to RAPID's objectives. *Random* replicates randomly chosen packets for the duration of the transfer opportunity. *Spray and Wait* restricts the number of replications of a packets to L, where L is calculated based on the number of nodes in the network. For our simulations, we implemented the binary *Spray and Wait* and set[2] $L = 12$. We implemented Prophet with parameters $P_{init} = 0.75$, $\beta = 0.25$ and $\gamma = 0.98$ (parameters based on values used in [74]).

We also compare RAPID to *Optimal*, the optimal routing protocol that provides an upper bound on performance. We also perform experiments where mobility is modeled as a power law distribution. Previous studies [37, 70] have suggested that DTNs among people have a skewed, power law inter-meeting time distribution. The default parameters used for all the experiments are tabulated in Table 3.4. The parameters for power law mobility model is different from the trace-driven model because the performance between the two models are not comparable.

For experiments using Dome traces, each data point is averaged over over 58 traces. Each of the 58 days is a separate experiment. In other words, packets that

[2]We set this value based on consultation with authors and using LEMMA 4.3 in [102] with $a = 4$.

	Power law	**Trace-driven**
Number of nodes	20	max of 40
Buffer size	100 KB	40 GB
Average transfer opp. size	100 KB	given by real transfers among buses
Duration	15 min	19 hours each trace
Size of a packet	1 KB	1 KB
Packet generation rate	50 sec mean	1 hour
Delivery deadline	20 sec	2.7 hours

Table 3.4. Rapid: Experiment parameters

are not delivered by the end of the day are lost. For experiments using synthetic mobility model, each data point is averaged over 10 runs with random seeds. In all of the experiments, Prophet performed worse than RAPID, *Spray and Wait*, and *MaxProp*, and therefore we omit Prophet results from the graphs.

3.6.2.2 Comparison with existing routing protocols

Our experiments show that RAPID consistently outperforms *MaxProp*, *Spray and Wait* and *Random*. We increased the load in the system up to 40 packets per hour per destination, when *Random* delivers less than 50% of the packets.

Figure 3.6 shows the average delay of delivered packets using the four protocols as we vary load. RAPID's routing metric is set to average delay (Eq. 3.1). When using RAPID, the average delay of delivered packets is significantly lower than *MaxProp*, *Spray and Wait* and *Random*. Moreover, RAPID also consistently delivers a greater fraction of packets as shown in Figure 3.7.

Figure 3.8 shows RAPID's performance when the routing metric is set to worse-case delay (Eq. 3.3) and similarly Figure 3.9 shows results when the metric is set to the number of packets delivered within a deadline (Eq. 3.2).

We note that among *MaxProp*, *Spray and Wait* and *Random*, *MaxProp* delivers the most packets, but *Spray and Wait* has marginally lower average delay than *Max-*

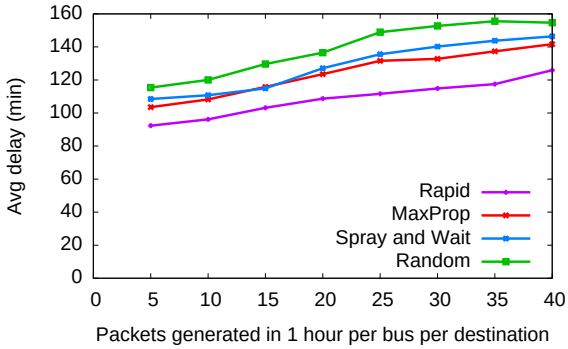


Figure 3.6. RAPID: Average Delay. RAPID has up to 20% lower delay than *MaxProp* and up to 35% lower delay than *Random*.

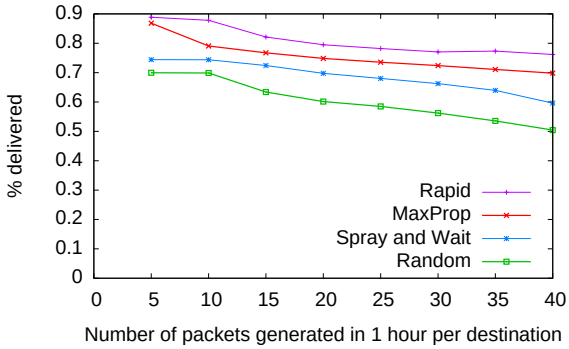


Figure 3.7. RAPID: Delivery Rate. RAPID delivers up to 14% more than *MaxProp*, 28% than *Spray and Wait* and 45% than *Random*.

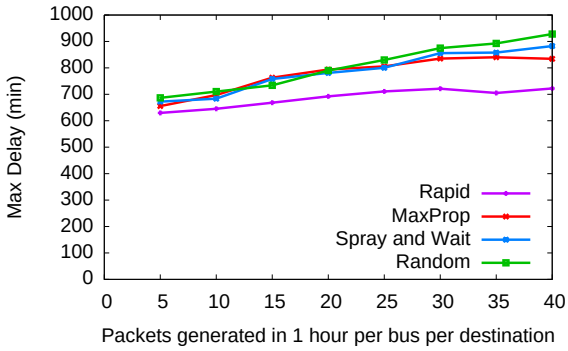


Figure 3.8. RAPID: Max Delay. Maximum delay of RAPID is up to 90 min lower than *MaxProp*, *Spray and Wait*, and *Random*.

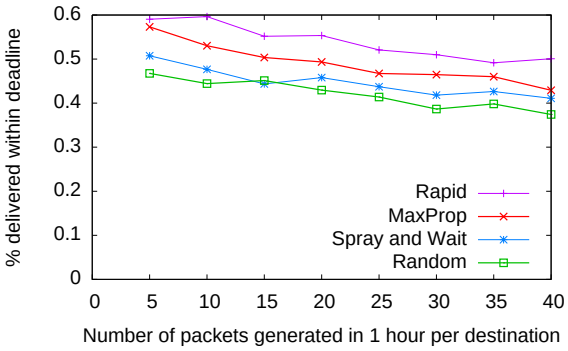


Figure 3.9. RAPID: Delivery within deadline. RAPID delivers up to 21% more than *MaxProp*, 24% than *Spray and Wait*, 28% than *Random*.

Prop. RAPID significantly outperforms the three protocol for all metrics because of its intentional design.

Standard deviation and similar measures of variance are not appropriate for comparing the mean delays as each bus takes a different geographic route. So, we performed a paired t-test [35] to compare the average delay of every source-destination pair using RAPID to the average delay of the same source-destination pair using *MaxProp* (the second best performing protocol). In our tests, we found p-values always

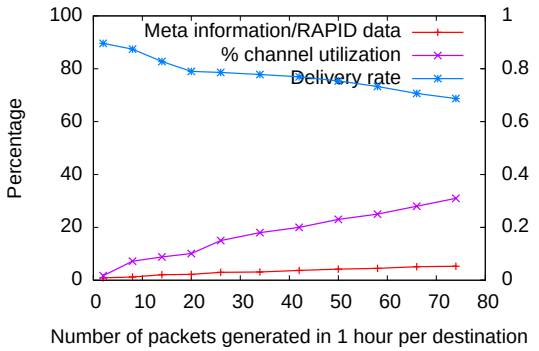


Figure 3.10. RAPID: Channel utilization. As load increases, delivery rate decreases to 65% but channel utilization is only about 35%.

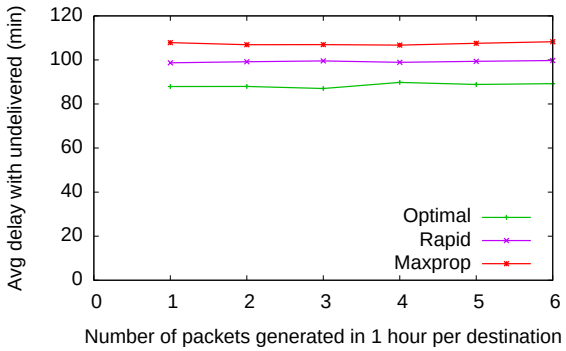


Figure 3.11. RAPID: Comparison with *Optimal*. Average delay of RAPID is within 10% of *Optimal* for small loads.

less than 0.0005, indicating the differences between the means reported in these figures are statistically significant.

3.6.2.3 Quantifying metadata overhead

We quantify the total metadata as a percentage of data. In particular, we increase the load from 5 to 75 packets per destination per hour to analyze the trend in terms of bandwidth utilization, delivery rate and metadata. Figure 3.10 shows this trend as load increases. The bandwidth utilization is about 35% for the load of 75 packets per hour per destination, while delivery rate is only about 65%. This suggests that the performance drops even though the network is under-utilized. This is because the available bandwidth varies significantly across transfer opportunities in our bus traces [31] resulting in bottleneck links.

We also observe that metadata increases to about 4% of data for high loads. This is an order of magnitude higher than the metadata observed as a fraction of bandwidth, again because of the poor channel utilization. The average metadata exchange per contact is proportional to the load and the channel utilization. However, metadata enables efficient routing and helps remove copies of packets that are already delivered,

increasing the overall performance of RAPID. Moving from 1-KB to 10-KB packets will reduce RAPID's metadata overhead by another order of magnitude.

3.6.2.4 Comparison with *Optimal*

We compare RAPID to *Optimal*, which is an upper bound on the performance. To obtain the optimal delay, we formulate the DTN routing problem as an Integer Linear Program (ILP) optimization problem when the meeting times between nodes are precisely know. The optimal solution assumes that the propagation delay of all links are equal and that node meetings are known in advance. We present a formulation of this problem in Appendix A.3. Our evaluations use the CPLEX solver [43]. Because the solver grows in complexity with the number of packets, these simulations are limited to only 6 packets per hour per destination. Jain et al. [62] solve a more general DTN routing problem by allowing packets to be fragmented across links and assigning non-zero propagation delays on the links, however, this limited the size of the network they could evaluate even more. Our ILP objective is to minimize the sum delays of all packets, where the delay of undelivered packets is set to time the packet spent in the system. Accordingly, we add the delay of undelivered packets when presenting the results for RAPID and *MaxProp*.

Figure 3.11 presents the average delay performance of *Optimal*, RAPID, and *MaxProp*. We observe that for small loads, the performance of RAPID using the in-band control channel is within 10% of the optimum performance, while using *MaxProp* the delays are about 22% from the optimal. RAPID using a global channel performs within 6% of optimal. The *global* control channel is an instant channel for exchanging metadata as opposed to the default (delayed) *in-band* control channel.One interpretation of the global channel is where all control traffic goes over a low-bandwidth, long-range radio such as XTEND [26].

3.6.2.5 Evaluation of rapid components

RAPID is comprised of several components that all contribute to performance. We ran experiments to study the value added by each component. Our approach is to compare subsets of the full RAPID, cumulatively adding components from *Random*. The components are *(i) Random with acks*: propagation of delivery acknowledgments; and *(ii)* RAPID-LOCAL: using RAPID but nodes exchange metadata about only packets in their own buffers.

Figure 3.6.3.1 shows the performance of different components of RAPID when the routing metric is set to average delay. From the figure we observe that using acknowledgments alone improves performance by an average of 8%. In our previous work, *MaxProp* [31], we show empirically that propagating acknowledgments clears buffers, avoids exchange of already delivered packets and improving performance. In addition, RAPID-LOCAL provides a further improvement of 10% on average even though metadata exchange is restricted to packets in the node's local buffer. Allowing all metadata to flow further improves the performance by about 11%.

3.6.3 Results from synthetic mobility models

Next, we use a power law mobility model to compare the performance of RAPID to *MaxProp*, *Random*, and *Spray and Wait*. When mobility is modeled using power law, two nodes meet with an exponential inter-meeting time, but the mean of the exponential distribution is determined by the popularity of the nodes. For the 20 nodes, we randomly set a popularity value of 1 to 20, with 1 being most popular. The mean of the power law mobility model is set to 0.3 seconds and is skewed for each pair of nodes according to their popularity.

3.6.3.1 Under varying load

Figure 3.13 shows the average delay for packets to be delivered (i.e., RAPID is set to use Eq. 3.1 as a metric). The average delay of packets quickly increase to 20

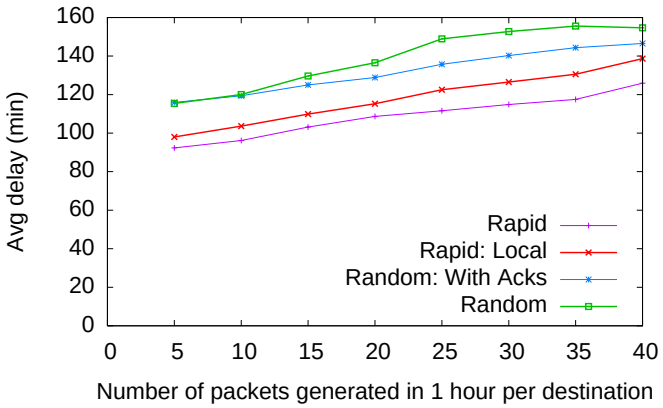


Figure 3.12. RAPID: Contribution of the different components to RAPID's performance

seconds as load increases in the case of *MaxProp*, *Spray and Wait* and *Random*. In comparison, RAPID's delay does not increase rapidly with increasing load, and is on an average 20% lower than all the three protocols.

Figure 3.14 shows the worse-case delay of packets when the load is varied (i.e., RAPID is set to use Eq. 3.3 as a metric). RAPID reduces worse-case delay by over 30% compared to the other protocols. For both the traces and the synthetic mobility, the performance of RAPID is significantly higher than *MaxProp*, *Spray and Wait*, and *Random* for the worse-case delay metric. The reason is *MaxProp* prioritizes new packets; older, undelivered packets will not see service as load increases. Similarly, *Spray and Wait* does not give preference to older packets. However, RAPID specifically prioritizes older packets to reduce worse-case delay.

3.6.3.2 Under varying buffer size

In this set of experiments, we varied the buffer size from 10 KB to 280 KB and compared the performance of the four routing protocols. We fixed the load to 20 packets per destination and generated packets with a inter-arrival time of 50 seconds.

Figure 3.15 shows how the average delay of all four protocols vary with increase storage availability. RAPID is able to maintain low delays even when only 10 KB space

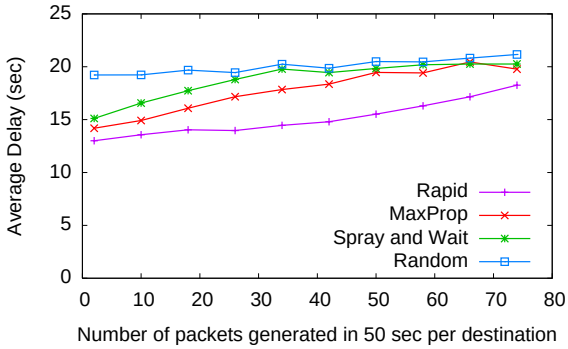


Figure 3.13. RAPID: Comparing average delay of routing protocols when nodes meet with power law distribution.

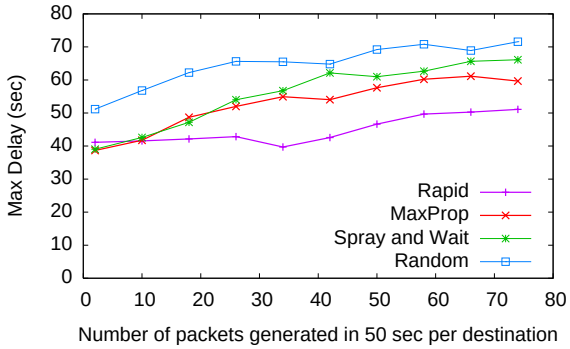


Figure 3.14. RAPID: Comparing worse-case delay of routing protocols when nodes meet with power law distribution.

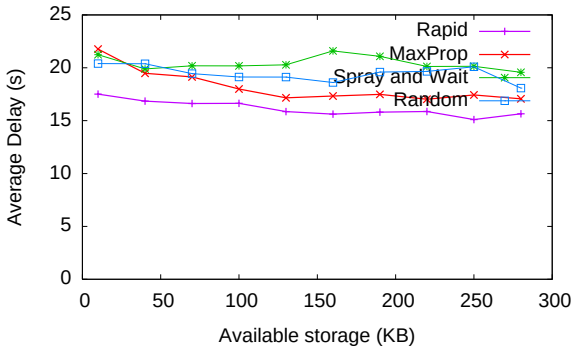


Figure 3.15. RAPID: Comparing average delay of routing protocols when nodes meet with power law distribution and when the buffer size is varied.

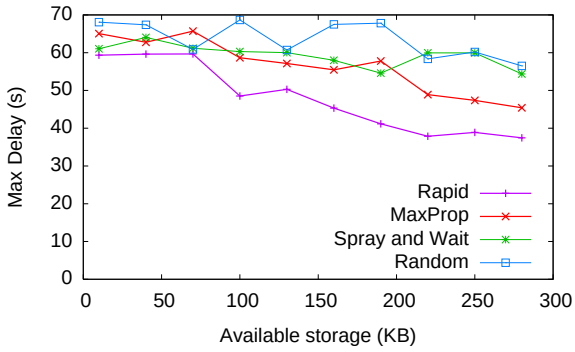


Figure 3.16. RAPID: Comparing worse-case of routing protocols when nodes meet with power law distribution and when the buffer size is varied.

is available at each node. In comparison, *MaxProp*, *Spray and Wait* and *Random* have an average 23% higher delay.

Figure 3.16 shows a similar performance trend in terms of minimizing worse-case delay. Similar to other experiments, the difference in performance between RAPID and the other three protocols is more marked for the worse-case delay metric.

Figure 3.17 shows how constrained buffers affect the delivery deadline metric. When storage is restricted, *MaxProp* deletes packets that are replicated most number of times, while *Spray and Wait* and *Random* deletes packets randomly. RAPID, when

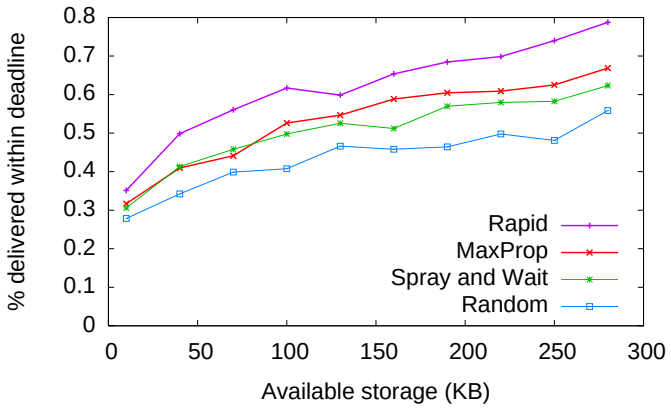


Figure 3.17. RAPID: Comparing delivery perfromance of routing protocols when nodes meet with power law distribution and when the buffer size is varied.

the metric is number of packets delivered within a deadline, deletes packets that are most likely to miss the deadline. RAPID is able to best manage limited buffers to deliver packets within a deadline and improves delivery performance by 12% compared to the second-best performing protocol. These experiments suggest that RAPID's utility-driven approach adapts well to storage restrictions as well.

3.7 RAPID Conclusions

In this chapter, I establish the thesis statement (Section 1.1) in the context of DTNs. Specifically, I show that disruptions in DTNs can be overcome by exploiting opportunistic replication. Replication involved routing multiple copies of the same packet, so that at least one of the copies reach the destination. When contact opportunity bandwidth is limited or when the load offered to the system is high, replicating multiple copies can cause a resource management challenge. RAPID addresses the challenge using a utility-driven protocol. Utility-driven RAPID only replicates packets if the marginal utility of replication outweighs the resources consumed.

I answer the research questions raised in Section 1.2 with respect to the DTN environment using measurement, protocol design, and testbed evaluation. Using a

measurement study on the Dome-DTN testbed, we characterize the performance of traditional wireless routing protocols in comparison to a naive replication protocol. We find that replication can significantly improve performance in disconnected environments. However, our measurement study shows that naive replication can hurt performance when resources are constrained, such as when the load offered to the network is high.

To solve the resource management challenge of replication, we designed a utility-driven routing protocol called RAPID. Rather than replicating indiscriminately, RAPID routes packets in the decreasing order of their marginal utility of replication. The protocol translates the routing metric to a per-packet utility function, and tunes its replications according to the specified routing metric and the available resources. The per-packet utility function is computed as probabilities, which is well-suited for the uncertain DTN environment.

We built and deployed RAPID on the Dome-DTN testbed and evaluate RAPID using both deployment and a trace-driven simulator. We show that the simulator provides performance results that are are within 1% of the real measurements with 95% confidence. We evaluate RAPID for three separate routing metrics: average delay, worst-case delay, and packets delivered within a deadline. Our experiments using trace-driven show that RAPID significantly outperforms four other routing protocols. For example, in trace-driven simulation experiments under moderate-to-high loads, RAPID outperforms the second-best protocol by about 20% for all three metrics, while also delivering 15% more packets for the first two metrics.

CHAPTER 4

THEDU: ENABLING WEB SEARCH IN INTERMITTENTLY CONNECTED NETWORKS

In an intermittently connected network, nodes are connected for a longer time compared to disconnected networks, but is not long enough to provide continuous connectivity. In this environment, it is not clear if moderately interactive applications can be supported when connectivity is intermittent. We look at this problem with respect to connectivity from WiFi APs. Supporting interactive applications using cheap WiFi access can pave the way for exploiting low cost WiFi for several applications including road monitoring [48] and traffic management [108].

In this chapter, I present Thedu [1], to answer the question: *Can moderately interactive applications such as web search be supported using intermittent WiFi access?* As a first step, I present a vehicular measurement study to characterize connectivity between moving vehicles and open WiFi APs. Based on the measurement study, we find that the primary challenge in adapting web search to an intermittently connected network is in tolerating disruptions. Currently, when Internet access is intermittent, Web search is cumbersome: a user issues a query, and a search engine returns a ranked list of URLs. Subsequently, the user clicks on one or more URLs and the search engine fetches the corresponding web page. When connectivity is intermittent, the web search process can stop during any one of the interactions, and the user needs to reissue the query.

[1] Thedu is the Tamil word for *search*. The first syllable rhymes with "hay" .

Instead, Thedu uses a proxy to aggressively prefetch web pages to transform the interactive web search process to a one-shot request/response process. The mobile client downloads the prefetched web pages from the proxy over a series of contact opportunities. Downloading all pre-fetched pages is inefficient and wasteful of the limited bandwidth available. In Thedu, we combine Information Retrieval (IR) techniques with a utility-driven prioritization algorithm to prioritize the downloaded web pages, so that the mobile client always downloads the most useful web pages.

The Thedu system is primarily designed to exploit *mobile-to-infrastructure (m2i)* contacts between a mobile client and an AP. As mobile nodes also come in contact with each other, allowing a DTN-like communication. We present an extension to Thedu to opportunistically exploit *mobile-to-mobile (m2m)* contacts in addition to m2i contacts. Exploiting m2m contacts can potentially improve performance in some scenarios such as kiosk-based rural networks, where the number of APs are limited and mobile nodes are plentiful in comparison.

The rest of this chapter is organized as follows: Section 4.1 describes the state-of-the-art in enabling applications in intermittently connected networks and Section 4.2 presents results from our measurement study. In section 4.3, I describe the Thedu system. In section 4.4, I describe the evaluation of Thedu and Section 4.5 concludes this chapter.

4.1 Related Work

Thedu draws upon ideas from a large body of prior work as described below.

Intermittently connected networks There appears to be a consensus on using a proxy-based architecture to tolerate disruptions in vehicular connectivity. Seth et al. [96] present an architecture and prototype for using vehicular mobility and stationary kiosks to extend the reach of sparsely deployed Internet gateways to a much larger rural area. They use an Internet proxy to hide disconnection from

legacy servers and pick a nearby proxy to improve TCP throughput. Ott et al. [86] use a proxy to hide disconnections and bundle a web page and inline objects into a single file to improve performance similar to HTTP 1.1 with pipelining. Hull et al. [61] present a mobile sensor computing system where vehicles sense and submit data to a central database that serves as a Web portal. Eriksson et al. [49] use a proxy to improve TCP throughput by distinguishing between losses in the wired and wireless halves of a connection. In comparison to these works, we build a proxy to aggressively prefetch web pages and enable search for mobile nodes.

Thedu is similar in spirit to the *infostation* [56] paradigm where base stations store information for mobile nodes, and a mobile node downloads the information when connected. Goodman et al. [56] quantify the benefit of this architecture and its incentive scheme based on an analytic model and synthetic traces for a file distribution application. Thedu has a complementary focus on interactive Web applications.

Measurement Zhang et al. [119] study inter-vehicular connectivity in the Dome [8] testbed and conclude that it is not always predictable. Others have conducted detailed studies of TCP performance [59] or link quality characteristics [110] under smaller controlled vehicular testbeds. The CarTel project [61] study the achievable throughput and meeting frequency of open WiFi APs in Boston from vehicles using a cab testbed. Our measurement study is similar to the CarTel study and is also based on open APs that are deployed organically, but focuses on disruption characteristics of vehicular networks, as opposed to performance characteristics.

Finally, aggressive prefetching has been used in several systems to mask disconnections, as discussed in Section 2.3.2. Thedu handles the resource management challenge of aggressively prefetching data using application domain knowledge.

Statistic	
Avg unique nodes:	151 (APs)
Number of contacts:	4964
Median contact duration (sec):	45
Total transfer (MB):	15, 071

Table 4.1. Thedu: Bus-AP meeting characteristics.

4.2 Measurement

We conduct a measurement study on the Dome-Infrastructure testbed described in Section 2.2. When the bus is within communication range of an AP, it associates to the AP and starts communicating with the AP. We log the connection duration, the disconnection duration, and the total throughput per bus-AP meeting over two weeks: from March 26–30, 2007 and from May 7–11, 2007.

Figure 4.1 shows the duration of bus-AP meetings during the two separate measurements; the two weeks show similar results. In both cases, about 80% of all meetings last for less than 50 seconds. Less than 5% of meetings last more than 400 seconds. These outliers result from buses that have powered-on but idle engines when in the garage (which has an AP) or buses that wait in traffic or at a bus stop while next to an AP. For both weeks, each set had hundreds of meetings with APs that lasted 10–180 seconds, as shown in Fig 4.2.

Figure 4.3 shows the bus-to-AP inter-meeting time. We ignore disconnections if the bus returns to the garage in the middle of the day. The median inter-meeting time is 5 minutes in the earlier week and 8 minutes in the latter week.

Table 4.1 tabulates per-day statistics collected from the trace data. The measurement study suggests that the bus-AP meetings can be used to transfer more than 15 GB of data in a day (Table 4.1). However, supporting interactive applications that require continuous access is difficult because buses get disconnected after a median contact duration of 30.17 seconds.

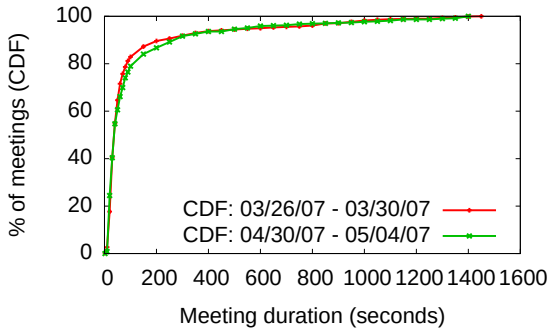


Figure 4.1. Thedu: CDF of Bus-to-AP meeting durations. Median of 45 seconds for both sets.

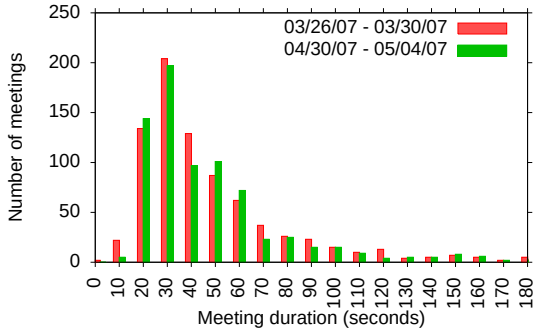


Figure 4.2. Thedu: Bus-to-AP interactions lasting less than 3 minutes.

4.3 Thedu Design

Web search and browsing are inherently interactive in nature, but do not have the stringent delay requirements of applications such as VoIP. However, these web applications cannot be supported today when connectivity is intermittent. Consider a tech-savy taxi passenger, Alice, who is visiting Springfield and wishes to find a sushi bar. Alice does not have a cellular data plan and instead takes advantage of the city's free WiFi. When within communication range of an AP, she sends the keywords "springfield sushi sake" to her favorite search engine. Before she can receive a response, the AP is out of range causing TCP to stall. She refreshes her browser to send the query again and receives a sorted list of Web links and snippets. She retrieves the top URL, but unfortunately it's for a restaurant in a Springfield in the wrong state, and while she is reading, the taxi has driven away from the AP. She waits for the next AP to retrieve the second URL or perhaps modify her query. Thedu seeks to improve user-perceived performance or robustness for interactive applications for exasperated vehicular Web users like Alice.

Thedu's architecture is guided by the following design goals.

- *Robustness*: Vehicular WiFi connectivity is prone to disruptions, so applications designed for persistent connectivity must be adapted to degrade gracefully.

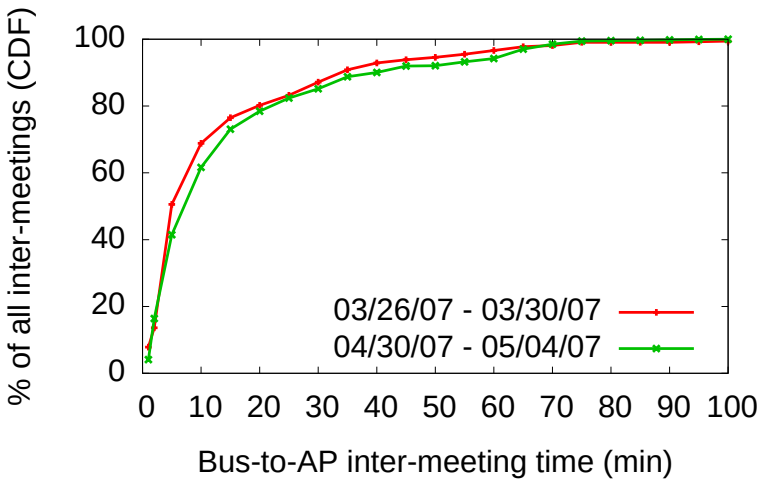


Figure 4.3. Thedu: CDF of bus-to-AP inter-meeting times.Median of 5 and 8 minutes for earlier and later set, respectively.

- *Low-cost deployability*: The system should be low-cost and need minimal change to existing infrastructure or software.

- *Extending access*: Many rural or developing regions have limited access points to the Internet, so the system should extend the Internet's reach.

The Thedu design uses three key ideas: *(i)* aggressive prefetching to transform Web search and browsing to a one-shot request response format; *(ii)* prioritizing web responses so that users download the most useful web pages first; and *(iii)* leveraging m2m contacts in rural and developing regions with limited Internet access.

4.3.1 Prefetching

Thedu uses a proxy to aggressively prefetch web pages from a search engine and transforms the Web search process to a one-shot request/response format. Figure 4.4 illustrates the Thedu system. A server-side proxy awaits a connection from a mobile. If the mobile has pending web pages, it downloads them immediately upon connection. In parallel, the proxy interacts with the Web search engine on behalf of the mobile to retrieve web pages for new queries. A similar proxy-based architecture

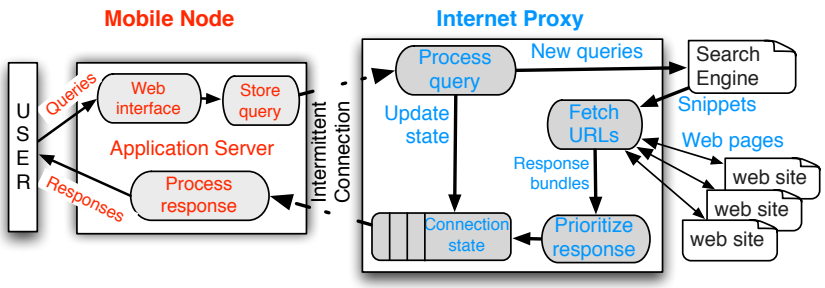


Figure 4.4. Thedu: System architecture.

has been used by several prior systems [61, 49] to make (non-interactive) applications more robust to disconnections.

The mobile accepts search queries from a user through a Web interface. The queries are sent to the proxy with a nonce that is used to re-identify the query after a disconnection. The proxy interfaces with a search engine, such as Google, Yahoo or MSN, and retrieves URLs and document snippets sorted by their relevance to the query. The proxy *prefetches* the web page bodies associated with the URLs in parallel. Studies have observed that 75% of users do not view more than the top 20 web pages and most users are interested in at most 5 results per query [64], so our implementation only prefetches the top 20 web pages. When the proxy receives multiple queries from a mobile, it retrieves web pages for each query in parallel. The proxy also retrieves all images and other objects and places them in a per-user queue. The mobile user then downloads the web pages from the queue.

4.3.2 IR meets networking

Our IR contribution is a novel set of techniques to process pre-fetched web pages, so that a mobile user downloads a significantly greater number of relevant web pages and avoids wasting bandwidth on web pages that are unlikely to be useful. Thedu prioritizes list of web pages in the decreasing order of their utility to the end-user. We

note that all search engines provide a ranked order of the web pages. We can use the same ranking for prioritization. Although simple, this strategy is not suitable because the rank cannot be used to prioritize web pages retrieved for different queries. Some search engines provide a more specific *relevance score* for each response to quantify the relevance of the web page to the query. However these scores are also not comparable across queries.

Thedu prioritizes web pages using two key ideas. First, it classifies the *query type* to determine user intent and how many relevant web pages the user desires. Second, it uses utility-driven prioritization to prioritize web pages across queries.

4.3.2.1 Query-type classification

Understanding the user's intent by classifying the query type can help increase the usefulness of web pages and limit useless prefetching. For example, returning only one web page suffices for a user's query of "Mobicom 2008", even if other web pages have a high relevance score. Broder [29] provides a nomenclature for classifying search queries as *homepage*, *content* and *service* queries. *Homepage* queries try to find known items where a single relevant response will satisfy the user; e.g., "cnn". *Content* queries seek an answer or a meaning and not a specific website and therefore require several relevant responses to satisfy the user; e.g., "kosovo conflict". *Service* queries are informational.

Thedu classifies queries as either homepage or non-homepage using a simple Bayesian classifier. For homepage queries, Thedu only returns one relevant response. If Thedu classifies the query as a non-hompage query, it returns the top 10 relevant responses.

We enumerate characteristic features of homepage and non-homepage queries and use these features to train the classifier. The features are summarized in Table 4.2. The features only depend on the URL, snippet, and title fields. We trained the

Homepage	Content
Query terms/acronyms occur in URL	Query is a question
All query terms occur in title and anchor text	One of the top-3 URLs is a *wiki*
Query is less than 3 words	Query is greater than 3 words
URL is a root	

Table 4.2. Thedu: Features used to classify the type of web query.

classifier on a set of queries using the URL, titles, and *snippets* fields for the top 10 web pages returned for each training query.

For example, for the training query "prime factors", the URL, title and *snippet* for the first response from the Google search engine is

- <url> `http://www.gomath.com/algebra/factor.php` </url>
- <snippet> To prime factor a number, begin dividing by the smallest possible prime and continue until the quotient is a prime number. </snippet>
- <title> prime factor </title>

We note that the URL, title, and snippet fields are short and easy to parse. Our technique is also independent of characteristics of the collection of documents. Therefore our classifier allow us to classify query-type without assistance from the search engine and as queries arrive.

Classifier Evaluation We analyzed the performance of the classifier on Google search engine and Indri [105] search engine. We trained the classifier using the queries from TREC[2]. We used a set of TREC queries to train the classifier and tested on a different set of TREC queries from 2001 TREC web-track.

When using Google, we retrieved the top 10 responses, URLs, and snippets for the training queries using the GoogleAPI (`code.google.com`). When using Indri, we

[2]NIST Text Retrieval Conference (TREC) is a research standard for evaluating Information Retrieval performance. TREC provides a collection of documents that the search engine operates on.

	Homepage	Content
Training size	45 queries	25 queries
Testing size	100 queries	25 queries
Accuracy	88%	73%

Table 4.3. Thedu: Classification results from Indri

queried the Indri search engine to retrieve the top 10 responses for the queries; Indri retrieved documents from the TREC collection.

The TREC database already separates queries into homepage and non-homepage queries. We evaluate the performance of our classifier by comparing against the corresponding TREC categorization. Table 4.3 presents the results of our classifier on Indri. When tested on Google, the accuracy of homepage prediction was 90% and the accuracy of non-homepage prediction was 71%. We note that using a simple set of features and training on a small set of queries and responses, our classifier is able to predict query type accurately.

4.3.2.2 Utility-driven prioritization based on relevance probabilities

Thedu prefetches several web pages for queries, but the client only has limited bandwidth to download them. Today's search engines do not prioritize web pages across different queries as they are typically not as constrained by bandwidth as intermittently connected networks. For example, is the fifth web response to query A with a relevance score of 4.3 more or less important than the seventh response to query B with a relevance score of 8.2? Thedu nodes must answer this question in order to prioritize web pages across queries, so that the web page with the highest utility to the user (or with the most relevance to the query) is downloaded first.

Thedu prioritizes web pages using two key ideas. First, it uses a *normalization* method so that relevance scores of web pages can be compared across queries. Second,

it computes the relevance probabilities of the normalized web pages, to prioritize the web pages according to their relevance to the query.

Query normalization We explain the query normalization technique as a modification of the Indri [105] search engine. Indri [105] is an academic search engine shown to be effective for web search. Indri returns a ranked set of documents in response to a query and associates each document with a relevance score. In case of web search, a document is a web page.

In the Indri model, the *relevance score* of a document D for query Q is estimated as

$$Score(D) = \prod_{w \in Q} \lambda P(w|D) + (1 - \lambda) P(w|C) \tag{4.1}$$

where λ is a smoothing constant, $P(w|D)$ is the probability of a word w occurring in a document D and $P(w|C)$ is the probability of the word w occurring in C, the entire collection of documents being searched.

Since the document score depends on the words in the query and the collection, the scores of documents for different queries (or over different collections) are not comparable. To normalize document scores across queries we propose estimating the relevance score using Kullback-Liebler divergence as follows:

$$Score(D) = \frac{1}{|Q|} \sum_{w \in Q} log \frac{\lambda P(w|D) + (1 - \lambda) P(w|C)}{P(w|C)} \tag{4.2}$$

This method normalizes scores so that document scores are comparable across queries and collection. Note that the document rank remains unchanged by normalization, as does the search effectiveness. In our deployment, we set $\lambda = 0.4$. A full derivation and explanation of Eq. 4.2 is available in a technical report [120].

Query prioritization Thedu computes relevance probability given the normalized relevance score, and prioritizes web pages according to the relevance probability. The relevance probability is computed as follows. Let $P(score|rel)$ denote the probability of score given the document is relevant and $P(score|nonrel)$ the probability

of score given the document is not relevant. Manmatha et al. [77] analyze the score distribution and suggest that scores of a relevant document, $P(score|rel)$, follows a gaussian distribution and scores for non-relevant documents, $P(score|nonrel)$, follow an exponential distribution. The relevance probability using Bayes' rule is

$$P(rel|score) = \frac{P(score|rel)P(rel)}{P(score|rel)P(rel) + P(score|nonrel)P(nonrel)} \tag{4.3}$$

We estimated the relevance probability of a document $P(r)$ using Eq. 4.3 by estimating the parameters for the Gaussian distribution ($P(score|rel)$) and Exponential distribution ($P(score|nonrel)$). We used a training data set of 25 content queries and 45 name-page finding queries from 2001 TREC web-track. We retrieved responses for the training data, classified them as relevant and non-relevant and used their relevance scores to determine the parameters of the gaussian and exponential distribution as $N(\mu = 0.669, \sigma = 0.00000517)$ and $exp(\lambda = 1.489)$.

4.3.3 Exploiting mobile-to-mobile contacts

Thedu leverages mobile-to-mobile (m2m) contact to route web pages when mobile nodes are within communication range of each other, using DTN-like communication. When routing using m2m contacts, the goal is to ensure that the relevant response is routed to the user before she disembarks from the vehicle. To this end, the goal is to maximize the number of relevant web pages delivered within a deadline.

As a first step, during a m2i contact with the proxy, mobile nodes download web pages that need to be routed to others, in addition to downloading their own web pages. The natural question is, If a mobile client has pending web pages, should it still download web responses for other peers? This notion is counter intuitive. Typically, in the DTN [17, 31, 102, 63] and MANET [90, 66] literature, the primary focus is to deliver a packet to its destination at the first available opportunity. However, for

Web search, we find that downloading peer responses, even when there are pending web responses for one-self, can provide substantial benefit. The reason is that the relevance probability is that responses have decreasing marginal utility and most of the utility resides in the first few web pages. Next, mobile nodes exchange web pages with each during node contacts. Below we describe m2m routing at the proxy and at the mobile node.

4.3.3.1 mobile-to-mobile routing at the proxy

The goal of Thedu's routing protocol is to maximize the number of relevant web pages delivered by a deadline. Let $P(r_i)$ be probability that web page r_i is relevant. Let $Q(r_i)$ be the probability that the web page r_i will be delivered within the deadline by the AP, and $Q_X(r_i)$ be the probability that the web page r_i will be delivered by node X within the deadline. if r_i is not destined to the node X, then the utility of routing r_i is probability of relevant responses delivered within the deadline = $P(r_i) \times Q_X(r_i)$; i.e., the probability that r_i is relevant to the query ($P(r_i)$)and the probability that r_i will be delivered within the deadline ($Q_X(r_i)$). If the web page r_i is destined to X, the utility of downloaded r_i is the probability that the relevant response will not be delivered within the deadline if r_i is not downloaded immediately, given by, $P(r_i) \times (1 - Q(r_i))$.

Web pages are routed using these utilities as follows: when X meets an AP, the proxy ranks two web pages r_i and r_j as shown in Figure 4.5. The proxy returns r_i if r_i has higher priority than r_j and vice versa.

In Step 1 of Figure 4.5, Thedu prioritizes responses for peer nodes according to the utility U. The utility r_i is defined as the probability that the web page is relevant and will be delivered within the deadline, i.e., the product of the relevance probability and the delivery probability. Between delivering a pending web page r_i and routing another peer's web page r_j (Step 2), Thedu does the following: if the pending web

Thedu(r_i, r_j):

1. *If r_i and r_j are both not destined to X*:
   (a) Set $U(r_i) = P(r_i) \cdot Q_X(r_i)$ and $U(r_j) = P(r_j) \cdot Q_X(r_j)$
   (b) Return web page with higher utility.
2. *If r_i is destined to X and r_j is not*:
   (a) If number of replications of $r_i = 0$, return r_i.
   (b) Else, set $U(r_i) = P(r_i) \cdot (1 - Q(r_i))$, $U(r_j) = P(r_j) \cdot Q_X(r_j)$.
   (c) Return web page with higher utility.
3. *If both are destined to X*: Send web page with higher relevance probability

Figure 4.5. Thedu: Prioritization at the proxy when leveraging m2m routing.

page r_i has never been routed though any peer node, then the client downloads the web page. If r_i has already been routed through a peer but is not yet delivered, Thedu estimates the utility of downloading r_i, which is the probability that the web response will miss its deadline. The utility of the other web page r_j is same as the first case. If both web pages are pending for X (Step 4), Thedu prioritizes the web pages according to the relevance probability.

4.3.3.2 mobile-to-mobile routing at the mobile node

Routing between two mobile nodes is also based on utility-driven prioritization. Suppose mobile node Y has an m2m contact with mobile node X. Node Y prioritizes all web pages to be delivered to X first, using the relevance probability. Y prioritizes the remaining web pages using the utility function

$$U(r_i) = P_r(r_i) \cdot Q_X(r_i) \tag{4.4}$$

Eq. 4.4 represents the probability that the web page r_i is relevant and will be delivered within the deadline by node X.

Both the prioritization protocol at the proxy as well as at the client use the probability that a web page is delivered within a deadline. To estimate this probability, the Thedu node estimates the expected meeting time with all other nodes as an average of past meeting times. Some nodes may never meet; in this case the expected meeting time is set to *infinity*. Thedu makes a simplifying assumption that the meeting times are exponentially distributed with the parameter equal to the expected meeting time. Accordingly, Thedu computes the probability that the web page is delivered within the deadline as $1 - e^{-\frac{1}{\lambda}d}$ where λ is the expected meeting time and d is the deadline. Approximating inter-meeting time to an exponential distribution simplifies delay computation and works well in practice [17].

4.3.3.3 Exploiting query locality

Thedu uses caching to further benefit from m2m contacts. Web queries are known to exhibit locality [116] and the query frequency follows Zipf's law. Thedu nodes exploit this domain knowledge by caching all web pages they route because the queries are likely to be requested by other users. On meeting a peer node, the node returns web pages from its cache if there is a hit. The proxy tracks the popularity of queries based on its frequency and marks each response web page with an indicator of the popularity of the query. This indicator allows the clients to remove web pages for unpopular queries from the cache, if necessary.

4.3.4 Scope and limitations

Several applications other than Web search can benefit from a design similar to Thedu to adapt to intermittently connected networks. For example, several commercial accelerators for Web browsing prefetch hyperlinks on a page to reduce user-perceived response time. Popular prediction algorithms use a Markov model [88] to estimate the probability that a user clicks on a hyperlink, which naturally serves as a utility in Thedu. Li et al. [73] present a utility-driven atmospheric sensing system

that uses progressive compression to send the most important data first under limited bandwidth, which is crucial for hazardous weather prediction applications like tornado detection. Similarly, layered-encoded multimedia can be streamed to vehicular users for buffered playback by prioritizing base layers.

The Thedu design cannot improve performance for interactive applications that do not permit prefetching. Our implementation of Thedu does not support dynamic content, mobile code, or interactive Ajax applications. Such applications require a more sophisticated proxy engineering which is outside the scope of this work.

4.4 Evaluation

We evaluate Thedu in two ways: *(i)* Using deployment on the Dome-Infrastructure testbed to compare the performance of Thedu with a state-less proxy that emulates todays web search engines and *(ii)* trace-driven simulations to compare the performance of Thedu with other variants and to quantify the benefits of m2m routing.

4.4.1 Deployment set up

We deployed Thedu and a *stateless* proxy, each for 5 days from March to April 2007. The two experiments differ as follows.

- The *Thedu proxy* is implemented as described in Figure 4.4.
- The *stateless proxy* strips features from the Thedu proxy so that it is equivalent to the case where mobile nodes do not use a proxy. The stateless proxy retrieves web pages for queries and returns bundles to the client in FIFO order (i.e., by completed retrieval time). It terminates all incomplete transactions when the client disconnects; retrievals begins anew upon reconnection.

Recall, from Section 4.3.1 that the Thedu proxy interfaces with an already available Web search engines (e.g., Yahoo, Google, or MSN) to retrieve the URLs of top

web pages, and then it prefetches the contents of the web pages. For our deployment we chose to use Indri [105], an academic Web search engine. An important difference between commercial search engines and Indri is that the latter indexes a large collection of static web pages; commercial search engines, on the other hand, actively crawl the web. Two features of Indri make it more appropriate for evaluating Thedu than a commercial search engine.

1. *Indri allows evaluation of retrieval performance:* The IR community and NIST's TREC [5] have built a standard web collection. The web collection has predefined user queries and human *relevance judgments* for each query; i.e., a list of web pages in the collection that are relevant to the query. Indri indexes this static web collection. The relevance judgment allows us to evaluate the performance of Thedu in terms of the number of relevant web pages delivered to the user. Note that the judgments are used *only* for evaluation and Thedu does not know *a priori* which web pages are relevant and which are not.

2. *Indri provides response scores:* Indri assigns a relevance score for the web pages. Thedu uses the relevance score to estimate the relevance probability, which is subsequently used to prioritize web pages. Commercial search engines use (non-normalized) relevance scores, but these relevance scores cannot be obtained by us. Instead, commercial search engines only return a ranked list of web pages.

The reason that we do not provide an interface from Thedu to a commercial API (e.g., Google) is that we would not be able to present evaluation results without a large study of user experiences. In contrast, by using Indri and TREC we control repeatable experiments that leverage past user studies of search relevance. In future work, we plan to deploy Thedu with the Google API using a simple heuristic to translate the rank to a relevance probability.

Collection:	W10g [5]
Number of TREC queries:	150
Queries:	TREC 2001
Search engine:	Indri
Query deadline:	30 min
Queries per hour:	10 per bus

Table 4.4. Thedu: IR parameters used for deployment.

We used Indri to index and store a standard collection of web pages from the TREC WT10G web collection [5]. We used standard queries from 2001 TREC web-track associated with the WT10G collection, for evaluation. We evaluate the retrieval performance using the relevance judgments. This query set and evaluation technique is commonly used in the IR community to measure retrieval effectiveness. We modified the Indri source code to provide normalized scores as described in Section 4.3.2.2. For both proxies, we removed queries and associated web pages after 30 min (since we assume most passengers would exit the bus by that point). The deployment parameters are tabulated in Table 4.4.

The queries associated with the W10G collection are limited and relevance judgments are only available for the TREC prescribed queries. The vehicular client periodically generates a *(queryID, query)* pair, where queryID is a monotonically increases sequence number and query is chosen from the predefined query set. The queryIDs allow a query to be repeated as if it were completely new, and allow us to evaluate performance for a large number of queries.

For both deployment experiments, queries are generated with an average of 10 per hour per bus with inter-arrival times drawn from an exponential distribution. At each AP opportunity, the bus connects to the proxy and sends all queries generated since the last connection. The buses periodically upload statistics of queries, web pages, and delays. Delays are calculated against each query's generation time stamp, so they include the time waiting for the next AP.

Statistics	Thedu Proxy	Stateless Proxy
Number of meetings	897	935
Total queries:	780	743
Total web pages returned:	5639	1207
Avg resps. per query:	7.2	1.6
Relevant web pages:	1630	401
Queries with at least 1 relevant web page:	529 (68%)	291 (39%)

Table 4.5. Thedu: Average per day network statistics during deployment.

4.4.2 Deployment results

The results of our deployments are presented in Table 4.5. Each result is a per-day average. Thedu was able to return more than 4.5 times as many web pages on average. More importantly, by prioritizing web pages according to the relevance probability, the number of relevant web pages sent by Thedu is 4 times larger than the stateless proxy. Finally, Thedu was able to send at least one relevant web page for twice as many queries compared to the stateless proxy. We measure the delay of receiving a relevant web page for a query. Figure 4.6 shows an empirical CDF of the delay of receiving relevant web pages at the client. We note that 90% of the time, the first relevant web page is received within 5 min. The mean delay is receiving the first relevant web page is 2.7 min and the mean delay in receiving all relevant web pages is 2.3 min. In the next section, we show that the average delay in receiving relevant web pages is a function of AP density.

4.4.3 Trace-driven simulation set up

We collected traces from the Dome-Infrastructure testbed between October 22 to November 18, 2007, which resulted in 20 days of trace data by excluding weekends and holidays. Each bus in the testbed constantly scans for open APs and other buses. If an AP is found, our software caches the DHCP lease to speed up the IP address acquisition process in the future. To determine if the connection is usable,

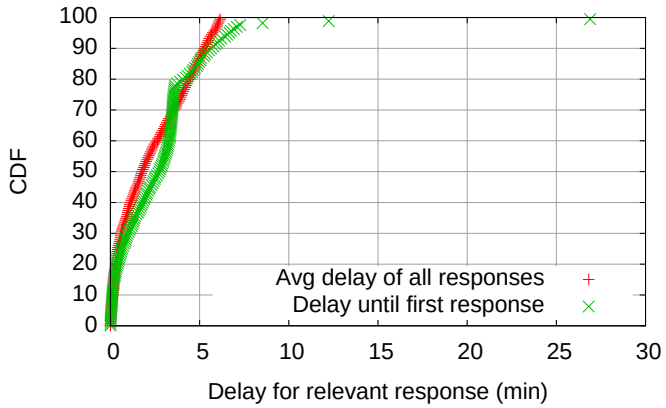


Figure 4.6. Thedu: CDF of the average delay in receiving relevant web pages during deployment.

the bus pings a known Internet server. If another bus is found, the connection lasts until the radios are out of range. During contacts, buses transfer random data, to measure the capacity of the connection. However, buses do not transfer data during m2i contacts to avoid overloading open APs operated by third parties; instead the buses ping a known server through the AP to estimate contact duration. The buses log information about both m2i and m2m contacts. The buses log the identity of the remote connection (SSID), the location and duration of contact, and for a m2m meeting, the number of bytes transfered. The logs are periodically uploaded to a central server. Table 4.6 tabulates the statistics of the collected trace data.

The experimental set up for the trace-driven simulations is similar to the deployment set up, described in Section 4.4.1. Each node simulates query generation and the inter-query time is drawn from an exponential distribution. The proxy retrieves the top 20 web pages from the Indri search engine. We compute the number of relevant responses delivered using the relevance judgement. The contact time and transfer size between two buses is given by the trace data. In the case of AP contacts, the trace provides contact duration but not the transfer size, because we do not send data during m2i contacts. We set the bandwidth for m2i contacts to 205 KBps, which is

Statistic	m2m	AP
Avg unique nodes:	21 (buses)	151 (APs)
Number of contacts:	242	4964
Avg contact duration (sec):	10.3	15.17
Avg bandwidth (KBps):	204.9	N/A
Total transfer (MB):	482	15, 071 (at 205 KBps)

Table 4.6. Thedu: The characteristics of m2i and m2m contacts.

the bandwidth of m2m contacts (see Table 4.6). The deadline for each query is set to 30 min.

4.4.4 Trace-driven simulation results

In this section, we present the trace-driven simulation results.

4.4.4.1 Thedu performance

We used the trace-driven simulation experiments to compare the performance of Thedu to two variants: *(i) Thedu without the query-type classification* that we describe in Section 4.3.2.1; and *(ii) Round-Robin* allocation of bandwidth for downloading responses. Last, we compare the performance of Thedu with a *stateless* proxy similar to our deployment.

Thedu without the query-type classification prioritizes responses based on relevance probability without considering the type of query (i.e., topic or homepage). The *Round-Robin* variation uses round-robin prioritization and allocates equal bandwidth to all queries. *Round-Robin* neither uses query-type classification nor prioritization.

Figure 4.4.4.1 shows the total number of web pages delivered as query load increases. The results are an average of 10 trials per point on the graph. The vertical lines at each point in the graph show the 95% confidence interval. The experiment

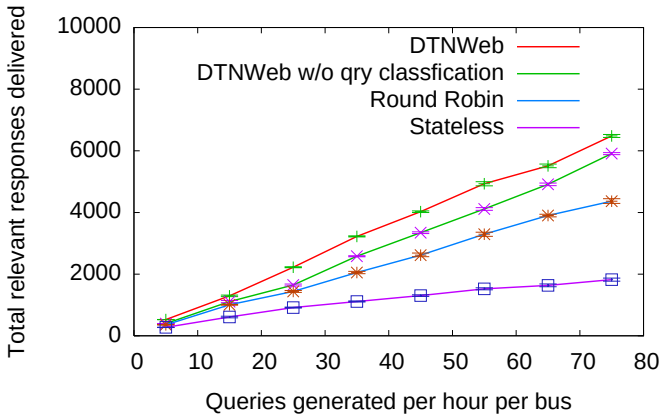


Figure 4.7. Thedu: Comparing the number of relevant web pages delivered using trace-driven simulations.

shows that Thedu increases the number of relevant web pages delivered by over 28% compared to a round-robin scheme. The experiments also show that the improvement in performance when using Thedu over *Round Robin* is statistically significant for loads greater than 5 queries per hour. Interestingly, we are only able to achieve this improvement when we predict the query-type with our classification algorithm; Thedu performs about 12% worse when it does not classify query-type. This is because, when Thedu does not use query-type classification, it returns several responses for a homepage query when only one is relevant, wasting bandwidth. Our results also indicate that maintaining state provides the most benefit in terms of performance. *Stateless* proxy performs about 40% worse than *Round Robin* and about 70% worse than Thedu.

4.4.4.2 Effect of AP density on Thedu

To understand the effect of AP density on Thedu performance, we divide the deployment region into grids, and study the delay of receiving relevant responses versus the AP density of a grid. The grid is centered at the Amherst town center and spans about 1 mile on each side. Though the Dome deployment spans a larger area, we concentrate on a snapshot of the deployment region that has a high frequency of

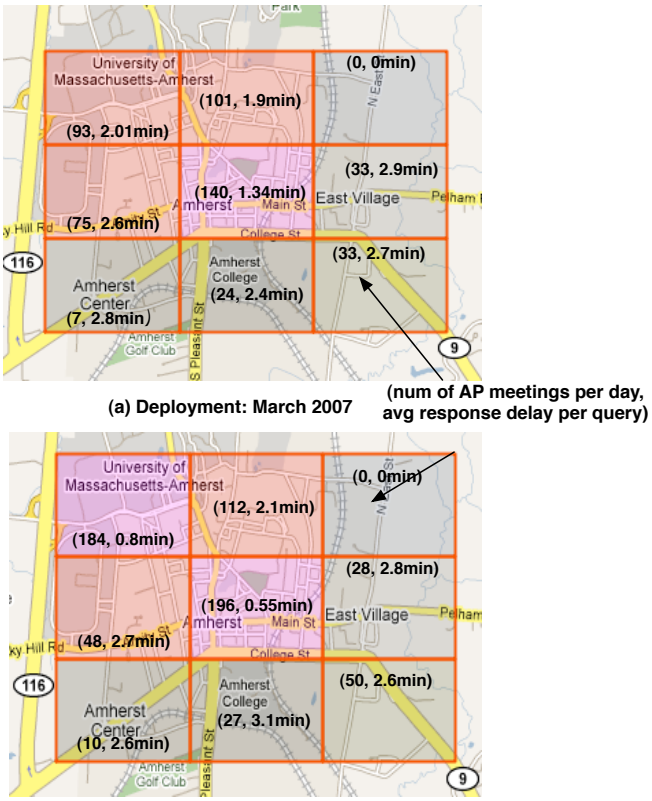


Figure 4.8. Thedu: Trends in AP density.

bus visits. We compare the deployment results with trace-driven simulations from traces collected in August 2007. In August 2007, a mesh network was added to the Dome testbed in the center of the town. Therefore, in the recent trace data, AP connectivity in the mesh area has increased, compared to during the deployment.

Figure 4.8 is a heat map of m2i contact frequency of the Thedu deployment in March 2007 and using 5 days of traces collected in November 2007. Buses meet APs more frequently in November for two grids compared to during the deployment: in the town (center grid) and near the university (top left grid). AP meeting frequency in the other grids does not change significantly between the deployment and trace. The meeting frequencies refer to per-day frequencies.

Next, we compute the average delays for responses delivered in each grid. In Figure 4.8, the average delay to deliver relevant responses is greater than 2.53 min in grids where the number of m2i contacts is less than 60. The delay reduces to less than 1.51 min when the number of m2i contacts becomes greater than 120. In the November traces, Thedu delivers relevant responses with a delay of 0.55 min in the Amherst town center. There are about 25 unique APs in the town center excluding the mesh. In a separate trace-driven experiment (not shown here), we find that the delay in receiving the first relevant response in the town center is 1.67 min. During the deployment, this delay was 2.11 min.

Though our measurement study shows that AP density has been increasing since March 2007, the trend may not be universal. In Dome, m2i meetings have increased partially due to the deployment of a mesh network.

4.4.4.3 Benefits of leveraging mobile-to-mobile contacts

In this section, we evaluate the benefits of m2m routing for our vehicular testbed using trace-driven simulations. Table 4.6 tabulates per-day statistics collected from the trace data. Access points are an inexpensive commodity product, and securing their access is harder than allowing public access. Hence, there were 151 open APs in our environment put up by third parties. Network-equipped buses are an expensive research platform, and the buses in our testbed were deployed by the authors only. Accordingly, our testbed has an imbalance. Given the APs found on the field, the ratio of m2m contacts to m2i contacts is 1:20. In terms of throughput, m2m routing can at most provide a benefit of $\frac{482}{15{,}071} = 3.2\%$ (refer to "total transfer" in Table 4.6). It is clear that even in a semi-urban area such as Amherst, m2m routing can provide little benefit.

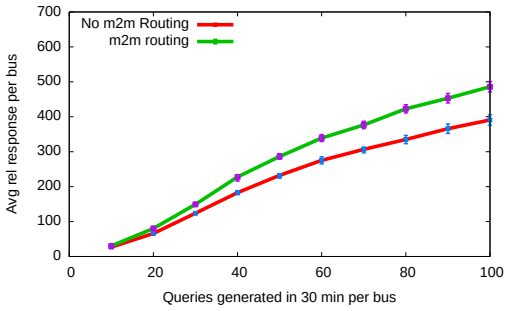


Figure 4.9. Thedu: Benefits of leveraging m2m contacts with 5 APs using trace-driven simulations.

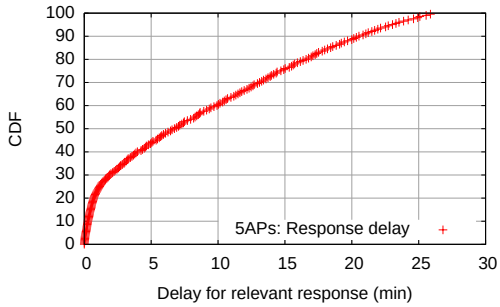


Figure 4.10. Thedu: Delay in receiving relevant responses with 5 APs using trace-driven simulations.

Instead, we evaluate the benefits of m2m routing when only a small number of APs are available. For example, in rural environments and OLPC [3]-like environments, AP deployment is sparse, but mobile laptop devices can help deliver web pages to clients that are multiple hops from the APs. In this experiment, we randomly chose 5 APs from the 151 APs available in the Dome testbed, and we find that the m2m-to-m2i contact ratio is 1:1.3, compared to 1:20 when 151 APs are available.

Figure 4.9 shows the number of relevant web pages delivered for all buses for varying load. In this setting, m2m routing provides up to 28% benefit over not using m2m routing. Figure 4.10 shows the CDF of the delay in receiving relevant web pages. The mean delay is 6.7 minutes, three times higher compared to when all APs were available for contact. In rural scenarios, even though m2m routing delivers more relevant web pages, the delays may deter users from using WiFi for interactive applications such as Web search. Therefore, our experimental results limit the applicability of m2m routing only to delay-tolerant applications even in rural areas with limited AP infrastructure.

[3]One Laptop Per Child

4.5 Thedu Conclusions

In this chapter, I establish the thesis statement (Section 1.1) in the context of intermittently connected networks. Specifically, I show that opportunistic prefetching can be exploited to enable moderately interactive applications such as web search in an intermittently connected network. Thedu uses opportunistic prefetching to converts the interactive web search process to a one-shot request/response process, more suitable for disruption-prone environments. However, aggressive prefetching introduces a resource management challenge of determining what data to prefetch. Thedu uses a utility-driven prioritization protocol to implement prefetching in the resource constrained environment.

I answer the research questions raised in Section 1.2 with respect to the intermittently connected environment using measurement, protocol design, and testbed evaluation. Using a measurement study on the Dome-Infrastructure testbed, we characterize disruptions in the testbed. We observe that vehicles are connected to an Internet connected AP for a median duration of 45 seconds, and are disconnected for a median duration of 8 minutes. In this environment, it is challenging to support the interactive web search application.

Instead, we design the Thedu protocol to overcome disruptions by translating the interactive web search process to a transactional process using aggressive prefetching. To determine what data to prefetch in the resource constrained environment, Thedu uses *(1)* a simple classifier that identifies queries that likely require only a single web response and *(2)* a utility-based prioritization protocol to normalize ranking of web pages across different search queries and determine the relative importance of the web pages. Thedu assigns a relevance probability to each web page to prioritize them. Finally, Thedu leverages m2m contacts in addition to the m2i contacts to improve web search in rural areas where connectivity is sparse.

We evaluate the performance of Thedu using a real deployment and using trace-driven simulation study on Dome-Infrastructure testbed. In our experiments, for areas with a high density of open APs, queries are answered in 0.55 minutes; the overall mean is 2.3 minutes. Moreover, compared to not using Thedu, users can expect a fourfold increase in the number of useful web pages retrieved in response to queries. Note that our results measure the time from querying to completed retrieval of a *relevant* web page, rather than simply the search engine results page or retrieval of the first, possibly irrelevant, result listed. Our trace-driven simulations show that in our testbed, where APs are densely deployed, m2m routing does not provide significant benefit compared to using m2i contacts alone. However, when buses are limited to using only 5 APs in our testbed to emulate a rural scenario, m2m routing provides up to 58% increase in the number of relevant web pages delivered per bus.

CHAPTER 5

VIFI: INTERACTIVE APPLICATIONS IN WELL-CONNECTED NETWORKS

There is an increasing number of WiFi mesh deployments that provide ubiquitous coverage, and in many cases, entire cities are being covered [113, 114]. The question we ask is: *can WiFi mesh deployments support highly interactive applications such as Voice over IP (VoIP) from moving vehicles?*. Highly interactive applications such as VoIP are challenging to support because they have strict requirements with respect to loss rates and connectivity. Supporting these applications at vehicular speeds will require highly efficient handoff policies as the vehicle moves from one mesh node to another.

In this chapter, I present ViFi, a protocol that supports interactive applications in WiFi meshes. As a first step, I present a measurement study to understand the challenges in supporting interactive applications in this setting. We find that with current WiFi handoff methods clients experience frequent disruptions in connectivity even when they may be close to WiFi AP. Handoffs in WiFi today are *hard*, i.e., at any given time, clients communicate with only one AP that is expected to offer the best connectivity. Instead, we find that using multiple APs simultaneously (i.e., exploiting overhearing by neighboring APs), can help reduce disruptions for vehicular clients.

Next I present the ViFi handoff protocol that reduces disruptions by leveraging opportunistic overhearing in wireless networks. The challenge in designing a protocol that exploits overhearing is in coordinating among APs to determine which AP should

forward the packet to the intended next hop. Without such coordination, multiple APs that opportunistically overhear the packet can transmit the packet, wasting resources. This coordination must be nimble enough to allow per-packet processing and must use the communication channel efficiently. ViFi addresses this challenge using a utility-driven sender prioritization protocol that assigns relaying probabilities to each node. APs that opportunistically overhear a packet but not its acknowledgment, relay the packet to the intended next hop according to their probability. The probabilities are computed by each node individually without need for per-packet coordination. The prioritization problem is formulated such that, collectively, wasted transmissions are minimized.

The rest of this chapter is organized as follows: In Section 5.1, I describe related work. In sections 5.2 and 5.3, I describe the measurement study and the ViFi protocol, respectively. In section 5.4, I describe the evaluation of ViFi and Section 5.5 concludes this chapter.

5.1 Related Work

Our work benefits from and builds upon a large body of work in wireless handoffs and opportunistic forwarding. We divide prior work into the following categories.

Using multiple APs

Distributed Radio Bridges [72], Divert [79], and MRD [80] all use multiple APs to improve client performance in enterprise WLAN deployments. The AP coordination mechanism in these systems assumes that a high-capacity LAN is available. For instance, in MRD, APs coordinate by sending all received frames to a central controller that is responsible for forwarding only one copy to the Internet. Thus, if clients typically reach three APs, the required LAN capacity is at least three times the cumulative sending rate of all clients. Because a high-speed backplane is typically not

available in mesh environments, the coordination mechanism of ViFi imposes little additional load on the backplane.

Network access from moving vehicles

Several early works (including our own), consider the problem of transferring data to individual APs as a vehicle drives by them, without maintaining a connection across APs [61, 58, 49, 55, 19]. They find that performance in this setting is severely hindered by overheads at several layers, such as DHCP and aggressive TCP backoffs due to losses, and propose methods to lower these overheads. We investigate the possibility of continuous connectivity across APs. We find that even if some of the overheads they observe (e.g., DHCP) are removed completely, the basic link layer connectivity remains problematic, especially for interactive applications.

Fast Handoffs There is a large body of work on minimizing the delay associated with handoffs in wireless networks [92, 16, 95, 28, 60]. This delay can be a major source of disruption in networks that otherwise have good wireless connectivity. Our work instead focuses on reducing packet losses during handoff, which can severely affect performance of interactive applications, even if the handoff delays are minimal.

Finally, related work on exploiting opportunistic overhearing in static mesh networks such as ExOR [27] and MORE [36] share similar goals but are not suitable for interactive applications, as described in Section 2.3.3.

5.2 Measurement

Our measurements are performed on the VanLAN testbed described in Section 2.2. We measure packet reception from a vehicle to every AP within communication distance. We then characterize the performance of applications, given the packet reception observed during measurement, using trace-driven studies. The goal of the measurement study is to characterize the performance of applications under different handoff policies.

5.2.1 Methodology

For the measurement study, each AP and vehicle broadcasts a 500-byte packet at 1 Mbps every 100 ms. We verified that self-interference of this traffic is minimal by comparing its packet reception ratio with the case where only one node sends at a time. Even though unicast packets, followed by acknowledgments, are more common today, broadcast packets suffice to probe the underlying connectivity and let us measure connectivity from a sender to all receivers simultaneously [15]. Nodes log all correctly decoded packets and beacons. Our results are based on analysis of traces collected over a two-week period.

We analyze the performance of different handoff policies using the collected trace. There are many possible handoff strategies; for a comprehensive evaluation, we study six different policies. Four of these are practical and are based on existing literature; the other two are idealized methods and lend insight into the inherent limitations of the practical policies. Our goal is to first understand the fundamental differences among the policies, so we ignore the time taken to switch between APs and the time to scan for APs. Research shows that careful design can tackle these issues effectively [92, 28, 60, 16].

We evaluate a handoff policy using these traces as follows. The policy determines which AP a client associates with at a given time. The client can communicate with only the associated AP when using a hard handoff policy. We assume that clients have a workload that mirrors our trace traffic; i.e., they wish to send and receive packets every 100 ms. The traces of broadcast packets and the current association determine which packets are successfully received.

We use two measures to provide insight into the performance of different kinds of applications under the different handoff policies: *(i)* aggregate performance; *(ii)* periods of uninterrupted connectivity. An aggregate performance measure considers the total number of packets delivered and the total time or distance over which the

vehicle is connected. These are relevant to delay or disruption-tolerant applications that care most about throughput during a large time interval, such as a few hours; e.g., synchronizing mail folders in the background. The period of uninterrupted connectivity measures contiguous time intervals when the performance of an application is above a threshold, for some definition of performance and threshold. This metric cares about performance at extremely short time intervals. Measuring periods of uninterrupted connectivity will, for example, tell us the length of time a VoIP caller can talk before the call quality drops. Applications such as instant messaging lie between these extremes; interpolating our results can provide insight into their performance.

The six handoff policies that we study are the following.

1. *RSSI*, where the client associates to APs with higher signal strength, measured as the exponential average of the RSSIs of received beacons. This policy is similar to what many clients, including the NICs in our testbed, use currently in infrastructure WiFi networks.

2. *BRR*, where the client associates to the AP with the highest exponentially averaged beacon reception ratio. This policy is inspired by wireless routing protocols that are based on the reception ratio of probes [42].

We use an exponential averaging factor of half for both methods above and find the results robust to the exact choice.

3. *Sticky*, where the client does not disassociate from the current AP until connectivity is absent for a pre-defined time period, set to three seconds in our evaluation. Once disassociated, the client picks the AP with the highest signal strength. This policy was used in the CarTel study [61].

4. *History*, where the client associates to the AP that has historically provided the best average performance at that location. Performance is measured as the sum of reception ratios in the two directions, and the average is computed across traversals

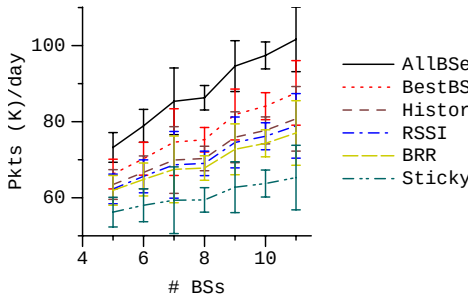


Figure 5.1. ViFi: Average number of packets delivered per day in VanLAN by various methods.

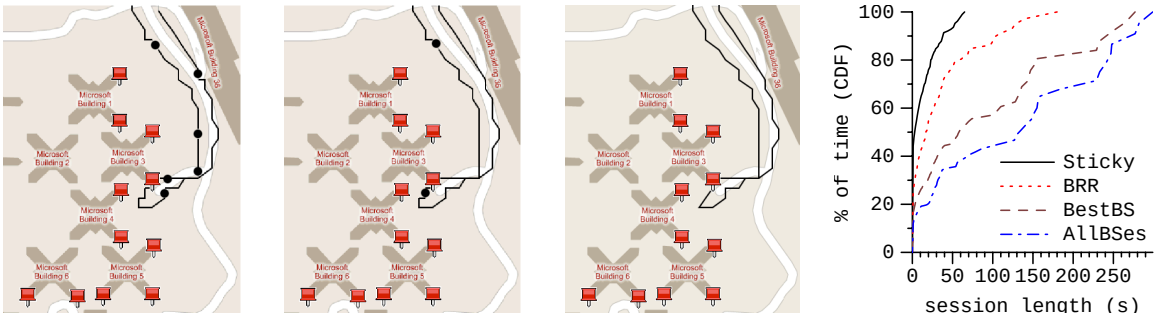
of the location in the previous day. MobiSteer shows the value of history in vehicular environments [83].

5. ***BestBS***, where at the beginning of each one-second period, the client associates to the AP that provides the best performance in the *future* one second. Performance is measured as the sum of reception ratios in the two directions. This method is not practical because clients cannot reliably predict future performance.

In cellular terminology, all of the policies above use *hard handoff* because the client associates with only one AP at a time. Using future knowledge, *BestBS* represents an upper bound on the performance of hard handoff methods.

6. ***AllBSes***, where the client opportunistically uses all APs in the vicinity. A transmission by the client is considered successful if at least one AP receives the packet. In the downstream direction, if the client hears a packet from at least one AP in an 100-ms interval, the packet is considered as delivered.

AllBSes is an ideal method that represents an upper bound on the performance of any handoff protocol. It exploits path diversity between the client and the set of nearby APs. Because of differences in CDMA and CSMA, it is not identical to, but is inspired by, macrodiversity methods in cellular networks [111]. Path diversity is known to improve performance in WiFi mesh and indoor infrastructure networks as well [27, 79]. We study if such benefits materialize in vehicular WiFi settings.



(a) *BRR* (b) *BestBS* (c) *AllBSes* (d) *Distribution of session length*

Figure 5.2. ViFi: (a)-(c): The behavior of three handoff methods for an example path segment in VanLAN. Black lines represent regions of adequate connectivity, i.e., more than 50% reception ratio in a one-second interval. Dark circles represent interruptions in connectivity. (d): The CDF of the time the client spends in a session of a given length.

5.2.2 Aggregate Performance Results

Figure 5.1 shows the packets delivered by the six handoff policies. To study the impact of AP-density, the independent variable in the graph is the number of APs in the system. There are eleven APs in VanLAN, and each point in the figure represents the average of ten trials using randomly selected subset of APs of a given size.

The graph shows that more packets are delivered as the density of APs increases but the relative performance of various methods is similar. *AllBSes* performs best, followed by *BestBS*, and then by *History*, *RSSI*, and *BRR*; the performance of *Sticky* is the worst. Ignoring *Sticky*, all methods are within 25% of *AllBSes*. This result suggests that for non-interactive applications, the choice of the exact method is not critical — however, results below demonstrate that interactive applications manifest great differences among the policies. Because of space limitations, we omit similar results for other aggregate performance metrics, such as the total time or distance for which the methods provide some minimal connectivity.

History, *RSSI* and *BRR* perform similarly for all measures that we study. The competitive performance of *History* confirms recent results [83] about the feasibility of using past experience to predict future performance. For visual clarity, we present results for only BRR as representative of all three in the remainder of this paper.

5.2.3 Uninterrupted Connectivity Results

To compare the ability of different handoff methods in providing uninterrupted connectivity, we start with a qualitative example. Figure 5.2 shows the behavior of *BRR*, *BestBS*, and *AllBSes* during one example trip of the vehicle. In this example, we define adequate connectivity to mean at least 50% of the packets are received in a one-second interval. Consistent with our aggregate performance measurement, each method provides adequate connectivity for roughly the same total path length. However, *BRR* contains several regions of inadequate connectivity. *BestBS* has fewer interruptions because it uses the optimal BS. *AllBSes* performs best as it uses multiple APs to further reduce the number of interruptions.

Frequent interruptions in BRR can be explained through a detailed analysis of the connectivity between a vehicular WiFi client and aAP. Contrary to earlier studies of controlled environments [87, 52], we find that in realistic environments this connectivity is often marred by *gray* periods where connection quality drops sharply. Gray periods are unpredictable and occur even close to APs. With BRR, the clients often find themselves experiencing a gray period with respect to the associated BS, which causes frequent disruptions in connectivity. But because they tend to be short-lived, gray periods do not severely impact aggregate performance. We have analyzed gray periods in our testbeds in more detail [23, 76] but omit that analysis from this paper.

Figure 5.2(d) quantitatively compares the handoff policies with respect to the cumulative time clients spend in an uninterrupted session of a given length. We see that the median session length of *AllBSes* is more than twice that of *BestBS* and

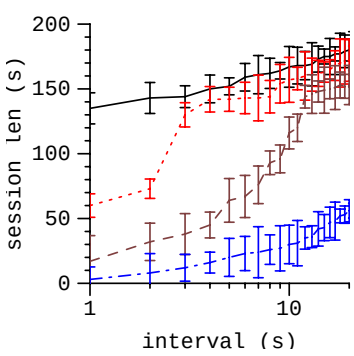


Figure 5.3. ViFi: The median session length in VanLAN as a function of the time interval and the minimum reception ratio.

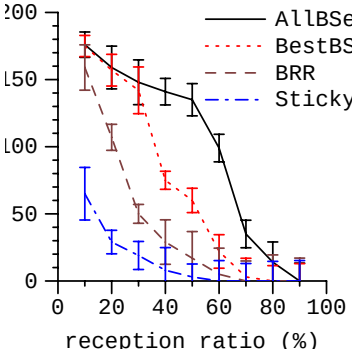


Figure 5.4. ViFi:The median session length in VanLAN as a function of the time interval used to define adequate connectivity.

more than seven times that of the more practical *BRR*. This suggests that a practical, multi-BS handoff policy can achieve significant gains over hard handoff.

To investigate how applications with different requirements can be supported, we now explore other definitions of adequate connectivity. Figure 5.3 varies the averaging interval while keeping the minimum reception ratio requirement fixed at 50%; Figure 5.4 varies the minimum reception ratio while keeping the averaging interval fixed at one second. A longer averaging interval represents less stringent requirements because the session is said to be interrupted only if there is no activity for a longer period. Similarly, a shorter reception ratio represents a weaker requirement. The results suggest that when the requirements are less stringent all methods other than *Sticky* perform similarly. But as the requirements become more demanding the relative advantage of using multiple APs increases. The right end of Figure 5.4 does not represent convergence but a degenerate point where the requirements are so strict that all methods have short sessions.

The above result suggests that for aggregate metrics, the choice of the exact handoff policy is not critical — however, the performance of interactive applications greatly vary depending on the handoff policy.

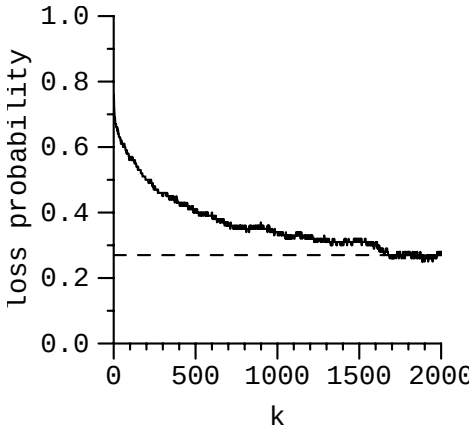


Figure 5.5. ViFi: Probability of losing packet i+k from an AP to vehicle given that packet i was lost.

Reception probabilities	
$P(A)$	0.75
$P(A_{i+1}\|\neg A_i)$	0.24
$P(B_{i+1}\|\neg A_i)$	0.57
$P(B)$	0.67
$P(B_{i+1}\|\neg B_i)$	0.18
$P(A_{i+1}\|\neg B_i)$	0.62

Table 5.1. ViFi: Unconditional and conditional packet reception probability from two APs to the vehicle.

5.2.4 Why is using multiple APs effective?

We now explain why *AllBSes* is significantly more effective than using only one AP even when that AP is judiciously chosen (as in the case of BestBS).

In the upstream direction, using multiple APs is effective because losses are roughly independent across APs and a packet sent by the vehicle is received by at least one AP with a high probability. In other words, the fact that a packet from the vehicle is lost at, for instance, the closest AP, has little bearing on whether it is lost at another AP. This independence of losses at receivers has been shown previously for outdoor WiFi meshes [27]. We find that it holds in our setting as well but omit detailed results.

Using multiple APs is effective in the downstream direction because it can tackle bursty losses better than single-AP systems [79]. Figure 5.5 shows evidence that losses are bursty in the vehicular setting. The figure plots the probability of losing the packet $(i+k)$ from an AP to vehicle in VanLAN given that packet i was lost. In this experiment, a single AP sends packets every 10 ms; we pick a different sending AP for each trip by the vehicle. The probability of losing a packet immediately after a loss is much higher than the overall loss probability. Thus, even when a vehicle is associated with an AP with a low average loss rate, it can lose many packets in a small time period, hurting interactive applications.

Leveraging overhearing by multiple APs helps overcome burst losses because when the vehicle is in a burst-loss phase with one AP a second AP may still be able to deliver packets to it. That is, most burst losses are path dependent (e.g., due to multipath fading) rather than receiver dependent. Figure 5.1 shows evidence that this holds for the vehicular environment and quantifies the effect for one pair of chosen APs in VanLAN. Each AP sends a packet every 20 ms. $P(A)$ and $P(B)$ are the unconditional downstream packet reception probabilities from APs A and B. $P(A_{i+1}|\neg A_i)$ is the conditional reception probability of receiving $(i+1)$-th packet from A given that the i-th packet from A was lost. Other probabilities can be similarly interpreted. We see that after a loss from an AP, the reception probability of the next packet from it is very low. But the second AP's probability of delivering the next packet is only slightly lower than its unconditional loss probability.

5.3 ViFi design

Motivated by the effectiveness of leveraging overhearing, we seek to develop a practical protocol that leverages AP overhearing to reduce disruptions. The key challenge is in coordinating among the APs such that the coordination scheme: *(i)* imposes minimal additional load on the inter-AP and vehicle-AP communication plane; *(ii)*

does not increase per packet latency, as that hurts interactive applications; *(iii)* can handle rapidly changing sets of APs.

Other works that leverage overhearing in WiFi networks either assume a high-speed inter-AP backplane [80, 79] or batch packets to amortize overhead [27, 36]. In our setting, however, a high-capacity backplane is often not available. We also cannot use batching because that increases latency for packets.

Our approach is to leverage opportunistic receptions by nearby APs, followed by utility-driven sender prioritization. Opportunistic receptions provide a low-overhead but unreliable means for disseminating information. With prioritization, each AP relays based on an independently computed relaying probability, which avoids the need for explicit coordination messages between APs. The resulting protocol is lightweight and decentralized. This protocol is similar in spirit to opportunistic scheduling protocols. A survey of opportunistic scheduling protocols for wireless networks can be found in [34]. The different scheduling protocols in literature are designed to provide fairness or to optimize throughput across nodes in the network. The goal of ViFi is to reduce disruption caused by dropped packets for a given set of source-destination pair.

In ViFi, the vehicle designates one of the nearby APs as the *anchor*. The anchor can be selected using any of the association methods that clients use today to pick an AP. The anchor is responsible for the vehicle's connection to the Internet—packets from the vehicle are forwarded through the anchor and packets from the Internet destined for the vehicle first arrive at the anchor. The vehicle designates other nearby APs as *auxiliary*.

The operation of ViFi is symmetric in both directions and is described below in terms of the source, *src*, and destination, *dst*, of the transfer. In the upstream direction, the vehicle is the source and the anchor is the destination. The roles are reversed in the downstream direction.

1. *src* transmits the packet P.
2. If *dst* receives P, it broadcasts an ACK.
3. If an auxiliary overhears P, but within a small window has not heard an ACK, it probabilistically relays P.
4. If *dst* receives relayed P and has not already sent an ACK, it broadcasts an ACK.
5. If *src* does not receive an ACK within a retransmission interval, it retransmits P.

ViFi can improve handoff performance because relaying by an auxiliary AP is better than a retransmission by the source itself (Section 5.2.4).

5.3.1 Computing relaying probability

The key challenge in computing the relaying probability for an auxiliary AP is to balance the trade-off between too few and too many relayed transmissions. With the former, the performance will degrade to that of hard handoff protocols; the latter will lead to excessive load on the vehicle-AP and inter-AP communication mediums.

The relay probability computation in ViFi is based on the following guidelines.

G1: Account for relaying decisions made by other potential relaying auxiliaries.

G2: Prefer auxiliaries with better connectivity to the destination.

G3: Limit the expected number of relayed transmissions.

The first two guidelines are easily motivated, but the third one is not immediately obvious. Should the number of relayed transmissions be low or be such that at least one of them reaches the destination? We use the former in ViFi, but we also considered a formulation based on the latter. We outline this formulation in Section 5.4.5, and show that it leads to too many relayed transmissions. Similarly, in Section 5.4.5, we study formulations that do not adhere to the other two guidelines and show that they do not perform well either.

Let $B_1, \cdots, B_K$ be the current set of auxiliary APs. Let node s be the source of a packet and node d be its destination, where *node* refers to either a vehicle or anchor AP depending on the packet's direction. Let p_{ab} denote the probability that b correctly receives a transmission from a, for $a, b \in \{s, d, B_1, ..., B_K\}$. ViFi estimates and disseminates the p_{ab} using periodic beacons (described in Section 5.3.3).

When some auxiliary B_x, $(x = 1, \cdots, k)$ hears a packet but not an acknowledgment, it uses its computed relaying probability to decide whether to relay. The overall strategy is to compute relaying probabilities so that the expected copies of the same packet relayed across all auxiliary APs equals 1. Within this constraint, auxiliary APs that are better connected to the destination are preferred.

We write the constraint on the expected copies of the same packet relayed using

$$\sum_{i=1}^{K} c_i r_i = 1 \tag{5.1}$$

Here c_i is the probability that auxiliary B_i is *contending* on this packet, that is, that B_i has heard the packet but not an acknowledgment, and r_i is B_i's relay probability. Strictly speaking, however, r_i is the number of times B_i should relay the packet. Except in pathological cases, r_i evaluates to less than one. We do not allow an auxiliary AP to relay a packet more than once.

We compute c_i using an approach described below. We then pick r_i satisfying Eq. 5.1 in a way that favors auxiliaries that are better connected to the destination node d. Specifically, we choose r_i such that

$$\frac{r_i}{r_j} = \frac{p_{B_i d}}{p_{B_j d}} \tag{5.2}$$

implying that $r_i = r \cdot p_{B_i d}$ for some r. Each contending auxiliary B_x solves Eq. 5.1 uniquely for r, and then relays the packet with probability $\min(r \cdot p_{B_x d}, 1)$.

A contending relay B_x computes c_i for each B_i, including itself, as the unconditional probability:

$$c_i = p_{sB_i}(1 - p_{sd}p_{dB_i}) \tag{5.3}$$

Here the first term, p_{sB_i}, is the probability that B_i receives the original packet, the second is the probability that B_i does not hear an acknowledgment. We have assumed that the two events are independent.

5.3.2 Salvaging

Sometimes a vehicle moves out of range before the anchor AP can deliver packets from the Internet. Application performance, especially that of TCP, can suffer if such groups of back-to-back packets are frequently lost.

To avoid this problem in ViFi, newly designated anchors *salvage* packets by contacting the previous anchor over the backplane. The new anchor learns the identity of the previous anchor from the beacons. Upon contact, the old anchor transfers any unacknowledged packets that were received from the Internet within a certain time threshold. We set the threshold to one second in our experiments, based on the minimum TCP retransmission timeout. The new anchor treats these packets as if they arrived directly from the Internet. Our salvaging mechanism is inspired by DTN routing and DSR [66], but it is based on pulling data rather than pushing.

5.3.3 Estimating packet reception probabilities

As WiFi APs do today, ViFi nodes send periodic beacons. The beacons are used to disseminate information about the packet reception probabilities needed by auxiliary APs, which include those between the other auxiliary APs and the anchor and between the other auxiliary APs and the vehicle.

A ViFi node estimates the reception probability from another node to itself using the number of beacons received in a given time interval divided by the number that must have been sent. Incoming reception probabilities are maintained as exponential

averages (α=0.5) over per-second beacon reception ratio. In their beacons, nodes embed the current incoming reception probability from all nodes that they heard from in the last interval. They also embed the packet reception probability from them to other nodes, which they learn from the beacons of those other nodes.

5.3.4 Retransmission timers

In the current 802.11 standard, acknowledgments are sent immediately after packet transmission, so the source knows when to retransmit an unacknowledged packet. But acknowledgments in ViFi may be delayed if they are generated in response to a relayed packet. The delay depends on the time for relayed packets to reach the destination, and thus retransmission timers must be set based on current network conditions.

The ViFi source sets the retransmit timer adaptively based on the observed delays in receiving acknowledgments. The source keeps track of the delays in receiving acknowledgments for its transmissions. The source then picks as the minimum retransmission time the 99th percentile of measured delays. Picking this high percentile means that sources err towards waiting longer when conditions change rather than retransmitting spuriously.

Transmission opportunities can arise for the source before the retransmission time for the earliest packet in the queue elapses. In such an event, instead of leaving the medium idle, the source sends the earliest queued packet that is ready for transmission. This can cause some amount of reordering when a later packet reaches the destination first. In our experiments, we find that the amount of reordering is small and does not hurt TCP performance. Hence, our current implementation does not attempt to order packets. If need be, it is straightforward to order packets using a sequencing buffer at anchor APs and vehicles.

5.4 Evaluation

Our evaluations are based on a deployment of ViFi on the VanLAN testbed and a trace-driven simulation based on traces collected from the Dome-Mesh testbed. While we deployed the APs in the VanLAN testbed and can modify the APs, we cannot modify the APs used in the Dome-Mesh testbed. Both testbeds are described in detail in Section 2.2.

We analyze a range of ViFi characteristics,

- Link layer performance of ViFi handoff
- Application performance when using ViFi handoff in comparison to using current 802.11 handoff
- Effectiveness of ViFi coordination and its efficiency with respect to medium usage
- Comparison with alternate formulations of the coordination problem

5.4.1 Methodology

Below we describe our implementation, trace-driven simulation set up and other experimental details.

5.4.1.1 Implementation

We have implemented ViFi on the Windows operating system. Almost all of our implementation sits in user space. A special in-kernel network driver intercepts packets from the OS and hands it to our process.

Our current implementation uses broadcast transmissions at the MAC layer because this lets us disable the automatic retransmission behavior of the NIC. Instead, a ViFi node sends it own acknowledgments for received packets. However, broadcast transmissions disable exponential backoff in response to losses which is intended to

reduce collisions. To reduce collisions, our implementation relies on carrier sense. The implementation also ensures that there is no more than one packet pending at the interface, to prevent a node from sending multiple back-to-back broadcast packets.

As an optimization, ViFi packets carry a 1-byte bitmap that signals which of the last eight packets before the current packet were not received by the sender. This helps save some spurious retransmissions of data packets that are otherwise made due to loss of acknowledgment packets.

Our implementation of ViFi was deployed on the VanLAN testbed for 2 months.

5.4.1.2 Trace-driven simulation set up

The trace-driven simulations are based on beacons logged by the buses in DomeMesh. The beacon loss ratio from an AP to the vehicle in each one-second interval is used as the packet loss rate from that AP to the vehicle and from the vehicle to the AP. This assumption ignores any asymmetry or finer-timescale behavior of packet loss. For inter-AP loss rates, we assume that AP pairs that are never simultaneously within the range of a bus cannot reach one another. For other pairs, we assign loss ratios between 0 and 1 uniformly at random. Our results are based on multiple trials and random seeds.

We use a QualNet-based implementation of ViFi to analyze performance. The loss rates are instantiated in the QualNet simulator by mapping them to the corresponding path loss values. This method allows us to program loss rates found in a real vehicular environment and therefore includes losses due to mobility and multipath fading, while still losing packets to events such as collisions.

Our experiments are based on a fixed 802.11b transmission rate of 1 Mbps to maximize range. We compare the performance of ViFi against *BRR*, the practical, hard handoff protocol. In BRR, client associates to the AP with the highest exponentially averaged beacon reception ratio.

We validate our trace-driven simulation method by collecting the same measurements from VanLAN and comparing its results to the deployment, i.e., we set the loss rate for each one-second interval to be the beacon loss ratio between the vehicle and the AP in that one second. Because we have inter-AP beacon loss ratios in VanLAN, unlike Dome-Mesh, we configure the inter-AS loss rates also as the inter-AP beacon loss ratio during each one-second interval.

5.4.1.3 Experimental set up

We compare the performance of ViFi against *BRR*, the practical, hard handoff protocol that we studied previously. To ensure a fair comparison, we implement *BRR* within the same framework as ViFi but with the auxiliary AP functionality switched off. Like ViFi, *BRR* uses broadcast transmissions without exponential backoff restrictions and uses bitmap acknowledgments. We omit experiments that show that *BRR* performs worse with unicast transmissions. The poor performance is because of backoffs in response to losses. In VoIP experiments, for instance, the length of disruption-free calls were 25% shorter.

Our experiments are based on a fixed 802.11b transmission rate of 1 Mbps to maximize range. Rate adaptation in vehicular networks is an open problem as current algorithms assume an environment that is less dynamic [58, 49].

Unless otherwise specified, results for VanLAN are based on at least three days of data for each protocol and workload configuration. For the Dome-Mesh simulations, we use traces from Channels 1 and 6. We profile each channel and log all received beacons. We profiled for 3 days in December 2007, during which the vehicle logged more than 100,000 beacons. The profiling channel was fixed so that beacons are not lost while scanning. We limit our analysis to BSes in the core of the town and to APs that are visible on all three days. There are 10 such APs on Channel 1 and 14 on Channel 6. About half of the APs on each channel belong to the town's mesh

(a) *Reception ratio=50%* (b) *Interval=1 sec*

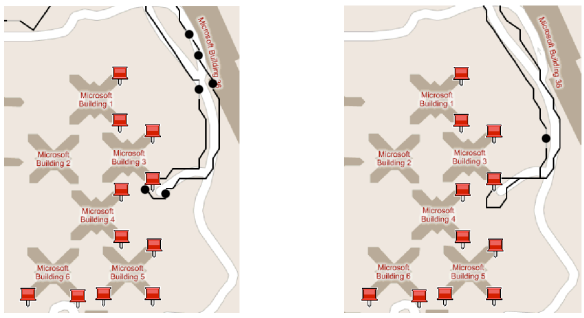
Figure 5.6. ViFi: The median session length in VanLAN as a function of the reception ratio threshold and time interval used to define adequate connectivity.

network and the rest belong to nearby shops. All errors bars in the graphs below represent 95% confidence intervals.

5.4.2 Link layer performance

We start by evaluating the basic link-layer connectivity provided by ViFi. This analysis is based on the VanLAN deployment and uses a methodology similar to that described in Section 5.2.

Figure 5.6 quantifies the performance of ViFi in comparison to the *BRR*, *BestBS*, and *AllBSes* handoff policies. The curves for *AllBSes* and *BestBS* are identical to those in Figures 5.3 and 5.4. In this experiment, the van and a remote computer attached to the wired network send a 500-byte packet to each other every 100 ms. Since we focus on basic link-layer quality provided by each protocol, link-layer retransmissions are disabled. The figure plots the median uninterrupted session length for various definitions of interruptions, as in Figures 5.3 and 5.4. The performance of ViFi is even better than *BestBS* and closely approximates *AllBSes*. It is notable that our simple and practical opportunistic protocol is able to beat the performance of the ideal single-AP protocol and approximate the ideal multi-AP protocol.



(a) *BRR* (b) *ViFi*

Figure 5.7. ViFi: The behavior of *BRR* and ViFi along a path segment in VanLAN.

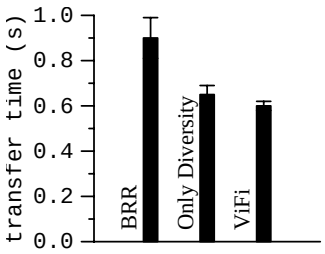


Figure 5.8. ViFi: TCP connection duration during VanLAN deployment.

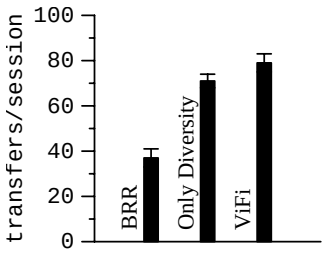


Figure 5.9. ViFi: Number of successful TCP transfers before disconnection during VanLAN deployment.

Figure 5.7 illustrates the behavior of *BRR* and ViFi, in a format similar to Figure 5.2. Black lines represent regions where the reception ratio was more than 50% in 1-second intervals. Dark circles represent interruptions. These are average case examples for the performance of these two protocols; individual runs differ. The paths are similar but not identical as they represent different days. We see that with *BRR* the path has several interruptions. ViFi performs significantly better, with only one interruption.

5.4.3 Application performance

We evaluate the performance of ViFi for two common interactive applications: short TCP transfers that are typical in web browsing and VoIP.

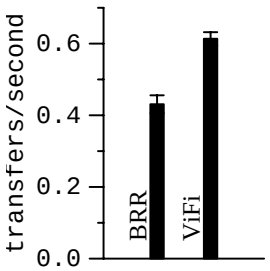


Figure 5.10. ViFi: TCP performance in Dome-Mesh Channel 1 in trace-driven simulations.

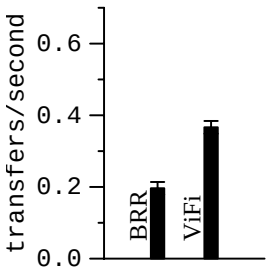


Figure 5.11. ViFi: TCP performance in Dome-Mesh, Channel 6 in trace-driven simulations.

5.4.3.1 Performance of TCP transfers

In this set of experiments, we focus on: *(i)* the time to complete a transfer; *(ii)* the number of completed transfers in a session, where a session is a period of time in which no transfer attempt was terminated due to a lack of progress. The vehicle repeatedly fetches a 10 KB file from a server connected to the wired network and the server does the same in the other direction. Transfers that make no progress for ten seconds are terminated and started afresh; we impose this limit because some transfers either hang or take a very long to complete due to packet losses at inopportune times in the TCP exchange.

Figure 5.8 shows the median time to complete a transfer. To isolate the benefits of leveraging overhearing and salvaging in ViFi, the middle bar shows the median time for a configuration in which probabilistic relaying was enabled but salvaging was disabled. The results show that ViFi's median TCP transfer time is about 0.6 seconds, which represents a 50% improvement over *BRR*. This improvement is higher than what would be predicted based on the number of additional packets delivered by the link layer (Figure 5.1). This brings out the difference between improvement in aggregate performance versus performance of interactive applications when using multiple APs to deliver the packet. The figure also shows that most of ViFi's gain results from leveraging overhearing, although salvaging does provide a noticeable gain

of about 10%. Figures 5.10 and 5.11 show the TCP performance over Dome-Mesh channels 1 and 6 respectively; ViFi significantly improves TCP performance in the Dome-Mesh testbed as well.

Figure 5.9 shows the average number of completed transfers per session. The average for ViFi is more than twice of *BRR*. Combined with its lower transfer times, this implies that users of ViFi will experience fewer disrupted transfers as well as better performance for individual transfers.

5.4.3.2 Performance of VoIP traffic

We evaluated the performance of VoIP sessions over ViFi by measuring the lengths of uninterrupted VoIP sessions. Supporting VoIP is more challenging than TCP because quality is sensitive to both loss and delay.

The industry-standard for evaluating a voice call is the *Mean Opinion Score* (MoS), which ranges from 1–5, with labels of perfect (5), fair (4), annoying (3), very annoying (2), and impossible to communicate (1). MoS is a perceptual measure, but it is commonly estimated from an R-factor score [41] as: 1, if $R < 0$; 4.5, if $R > 100$; and $1 + 0.035R + 7 \times 10^{-6}R(R-60)(100-R)$, otherwise. R-factor is sum of four terms $R = 100 - I_s - I_d - I_{ef} + A$, where I_s is the signal-to-noise impairments, I_d and I_{ef} are impairments due to delay and loss, and A is expectation factor, which is higher when users expect lower quality. The impairments are functions of the codec.

We use the G.729 codec, which is implemented on most VoIP devices. For simplicity, we set A to zero (though it may be higher given the challenging environment). Then, the R-factor reduces to [41]: $94.2 - 0.024d - 0.11(d-177.3)H(d-177.3) - 11 - 40log(1+10e)$, where d is the mouth-to-ear delay which includes the coding delay, network delay, and the delay introduced by the jitter buffer, e is the total loss rate which includes losses in the network and losses due to late arrivals, and H is the Heaviside step function: $H(x) = 1$ if $x > 0$; 0 otherwise.

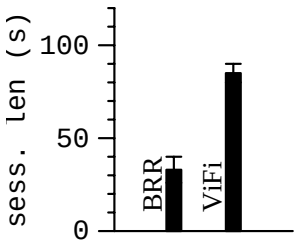


Figure 5.12. ViFi: Median length of uninterrupted VoIP sessions during VanLAN deployment.

Per the codec, we generate 20-byte packets every 20 ms. Following convention, we assume that the coding delay is 25 ms, the jitter buffer is 60 ms, and that the wired segment of the end-to-end path adds 40 ms (corresponding to cross-country paths in the USA) to each VoIP packet. Aiming for a mouth-to-ear delay of 177 ms (because the impairment due to delay increases significantly beyond that) means that packets that take more than 52 ms in the wireless part should be considered lost. We measure one-way delays by applying piecewise linear regression [81] to remove clock skew and assuming that the minimum one-way delay is identical in the two directions.

We quantify the VoIP performance using the median length of time between interruptions. We deem an interruption to have occurred when the MoS value drops below 2 for a three-second period. Three seconds is roughly the time it takes to enunciate a short English sentence and a MoS value lower than 2 represents a severe disruption in call quality.

Figure 5.12 shows the results for our deployment on VanLAN. Because our results (not shown here) indicate that salvaging brings little benefit for VoIP, we do not isolate overhearing and salvaging components of ViFi in the figure. The results show that the average session lengths are much longer with ViFi: the gain is over 100%. Figures 5.13 and 5.14 show that ViFi improves VoIP performance over 50% in Channel 1 of Dome-Mesh, and over 65% in Channel 6 of Dome-Mesh. We also find that the overall call

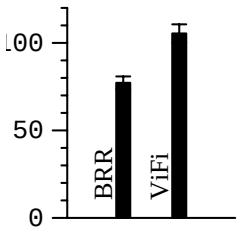


Figure 5.13. ViFi: VoIP performance in Dome-Mesh Channel 1 in trace-driven simulations.

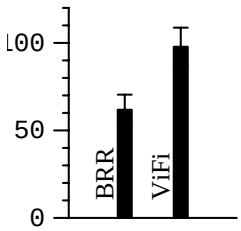


Figure 5.14. ViFi: VoIP performance in Dome-Mesh, Channel 6 in trace-driven simulations.

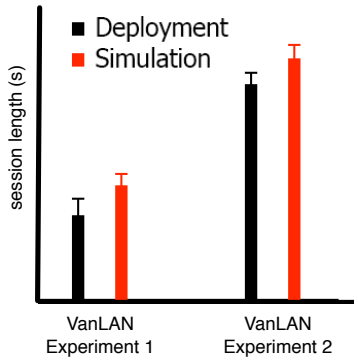


Figure 5.15. ViFi: Comparing VoIP performance between trace-driven simulation and VanLAN deployment.

quality with ViFi is better as well. The average of three-second MoS scores is 3.4 with ViFi and 3.0 with BRR. Thus, our results show that users of ViFi experience better call quality and significantly fewer disruptions in their voice calls.

Finally, we validate our simulator by comparing results of our deployment with trace-driven experiments using VanLAN traces using two different days of trace data. Figure 5.15 shows that the VoIP session lengths in the simulation come within 5 seconds of the observed session length for experiments done on two different days.

		Upstream	Downstream
A1	Median number of auxiliary APs	5	5
A2	Average number of auxiliary APs that hear a source transmission	1.7	3.6
A3	Average number of auxiliary APs that hear a source transmission but not the acknowledgment	0.6	2.5
B1	Source transmissions that reach the destination	67%	74%
B2	Relayed transmissions corresponding to successful source transmissions (i.e., false positives)	25%	33%
B3	Average number of auxiliary APs that relay when a false positive relay occurs	1.5	1.5
C1	Source transmissions that do not reach the destination	33%	26%
C2	Cases where at least one auxiliary AP overhears a failed source transmission	66%	98%
C3	Cases where zero auxiliary APs relay a failed source transmission (i.e., false negatives)	10%	34%
C4	Relayed packets that reach the destination	100%	50%

Table 5.2. ViFi: Detailed statistics on the behavior of ViFi during VanLAN deployment.

5.4.4 Analyzing ViFi

5.4.4.1 Effectiveness of coordination

In this section, we present detailed statistics on the behavior of ViFi to provide insight into the effectiveness of its coordination mechanism. Table 5.2 shows data from the TCP experiments in VanLAN. Row B2 shows that ViFi has few false positives. We define false positives as the number of additional relayed packets that are relayed to the destination. Comparing it with the average number of auxiliary APs that receive the source transmission (Row A2), we can infer that the coordination mechanism of ViFi is effective at curtailing unnecessary relaying. If every auxiliary AP that overheard a packet relayed the packet, the false positive rate would have been 170% and 360% in the upstream and downstream direction, respectively. If auxiliary APs deterministically relayed whenever they hear source transmission but not an acknowledgment, the false positive rate would have been 60% and 250% (Row A3). In

other words, relying on overhearing acknowledgments is not sufficient to curtail false positives; probabilistic relaying is needed as well.

Row C2 shows that auxiliary APs often overhear packets that do not reach the destination during the source transmission. Row C3 shows that in such cases, the false negative rate is low. We define false negative rate as the number of times no auxiliary relays a failed transmission divided by the number of failed source transmissions. Combining the two rows, we can infer that roughly 65% of the lost source transmissions are relayed in each direction.

5.4.4.2 Efficiency of medium usage

The beter application performance of ViFi does not stem from it using the medium more aggressively; in fact, its overall efficiency is comparable to that of *BRR*. We measure efficiency as the number of application packets delivered per transmission, in the channel between the vehicle and the APs.

We compare ViFi with *PerfectRelay* and *BRR*. In the *PerfectRelay* protocol, exactly one basestation relays only if the intended destination did not hear the packet. We estimate its efficiency using packet-level logs of ViFi. We use the logs collected during the measurement study. In the upstream direction, a packet is considered delivered by *PerfectRelay* if at least one AP hears it. In the downstream direction, a complication is that even if an AP relays the packet, the vehicle may not hear it. We get around this by: *(i)* assuming that the outcome of the relaying is identical to that of ViFi if at least one of the APs relayed the packet; and *(ii)* the relaying is successful if no AP relayed it in ViFi.

Figure 5.16 shows the upstream efficiency. We see that the efficiency of ViFi is better than *BRR* and nearly as high as *PerfectRelay*. Figure 5.16 shows the downstream efficiency. For downstream, all three protocols have similar efficiency. *BRR*

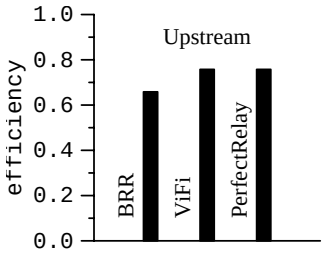


Figure 5.16. ViFi: Efficiency of medium usage for upstream communication.

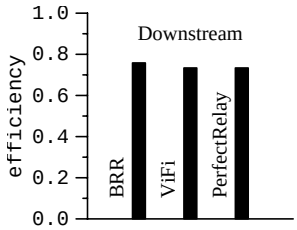


Figure 5.17. ViFi: Efficiency of medium usage for downstream communication.

has slightly better efficiency because in ViFi the AP chosen to relay a packet may be distant.

5.4.5 Comparison with other formulations

We compare ViFi's coordination mechanism with three other formulations. Each formulation violates one of the three guidelines outlined in Section 5.3.

¬G1: Auxiliary BSes ignore the presence of other potential auxiliary BSes. Each relays with a probability equal to its delivery ratio to the destination.

¬G2: Auxiliary BSes ignore loss rate to the destination. Each relays with a probability equal to $\frac{1}{\sum_i c_i}$, where c_i is that the auxiliary BS i is contending (Eq. 5.3)

¬G3: Auxiliary BSes relay such that the expected number of packets *received* by the destination is 1. (Recall that in ViFi, the expected number of packets *relayed* is 1.) Within this constraint, the objective is to minimize the number of relays. This formulation is an optimization problem: $\min \sum_i r_i \cdot c_i$ subject to $\sum_i r_i \cdot p_{B_i d} \cdot c_i >= 1$.

An optimal solution to this optimization problem, is $r_i = 0$ if $s_i > 1$; $r_i = 1$ if $s_i + p_{B_i d} \cdot c_i < 1$; and $r_i = \frac{1-s_i}{p_{B_i d} \cdot c_i}$ otherwise; where $s_i = \sum_{j: p_{B_j d} \geq p_{B_i d}} p_{B_j d} \cdot c_j$. In simpler terms, this solution first picks the auxiliary BS_x with the highest $p_{B_x d}$ and sets r_x=1. It stops if that satisfies the constraint above. Otherwise, it picks BS_y with the next highest $p_{B_y d}$. If the constraint is satisfied by r_y=1, then y

	ViFi	$\neg G1$	$\neg G2$	$\neg G3$
False positives	19%	50%	40%	157%
False negatives	14%	14%	12%	10%

Table 5.3. ViFi: Comparison of different downstream coordination mechanisms for DieselNet Ch. 1.

relays with a probability $\frac{(1-p_{B_x d} \cdot c_x)}{p_{B_y d} \cdot c_y}$. Otherwise, r_y is set to 1, and the BS with the third highest $p_{B_i d}$ is picked, and so on.

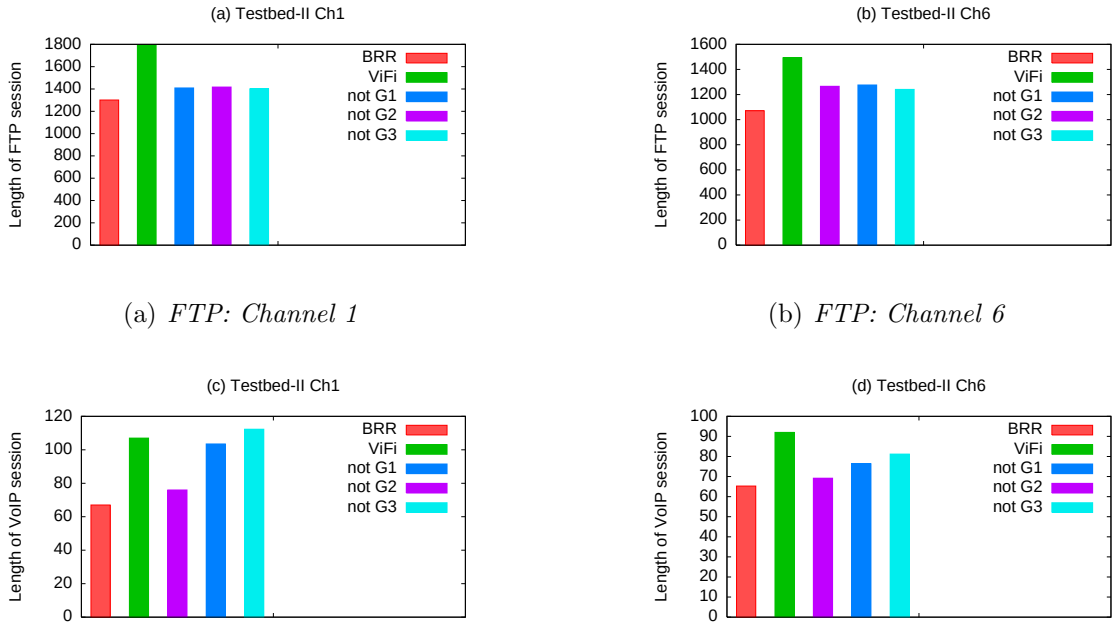
We find that compared to these other schemes ViFi strikes a good balance between false positives and false negatives. Table 5.3 shows the results for simulations over DieselNet's Channel 1 environment. While the false negatives for all schemes are roughly similar, ViFi has substantially lower false positives. Further, we observe in our experiments that the number of packets saved by $\neg G2$ is a lot lower than ViFi and that the false positive rate of $\neg G1$ increases rapidly with the number of auxiliary BSes.

Figures 5.18(a) - (b) show that the performance of the TCP application of all three alternate schemes is worse than that of ViFi. However, in Figures 5.18(c) and (d), the VoIP performance of $\neg G1$, $\neg G3$ is similar to that of ViFi for channel 1. Both $\neg G1$ and $\neg G3$ are aggressive protocols that relay several packets at the expense of congesting the medium. The TCP application performance appears to be significantly affected by congestion, unlike VoIP.

5.4.6 Findings from parameter-driven simulations

To understand the impact of environmental factors that we cannot control in testbed experiments, we conduct parameter-driven simulations using Qualnet.

For the QualNet experiments, the node's were placed randomly and the propagation pathloss model was set to the two-ray model. Figures 5.19(a) - (d) compares the



(a) *FTP: Channel 1*

(b) *FTP: Channel 6*

(c) *VoIP: Channel 1*

(d) *VoIP: Channel 6*

Figure 5.18. ViFi: Comparison of application performance between BRR, ViFi and three alternate formulations.

performance of ViFi and *BRR* for varying BS density and speed. Our main findings are:

1. Across all BS density levels that we study, ViFi performs significantly better than *BRR* for both TCP and VoIP experiments. Its relative benefit is highest for higher density levels. For these experiments, the speed was set at 30Kmh.

2. Across all speed levels that we study, ViFi performs significantly better than *BRR*. While speed does not affect the TCP experiments, higher speeds decreased the session length of VoIP. We fixed the number of BSes to 8. In a separate experiment, we also find that at low BS densities, it is better for vehicles to drive faster as they are able to get out of regions of poor coverage faster.

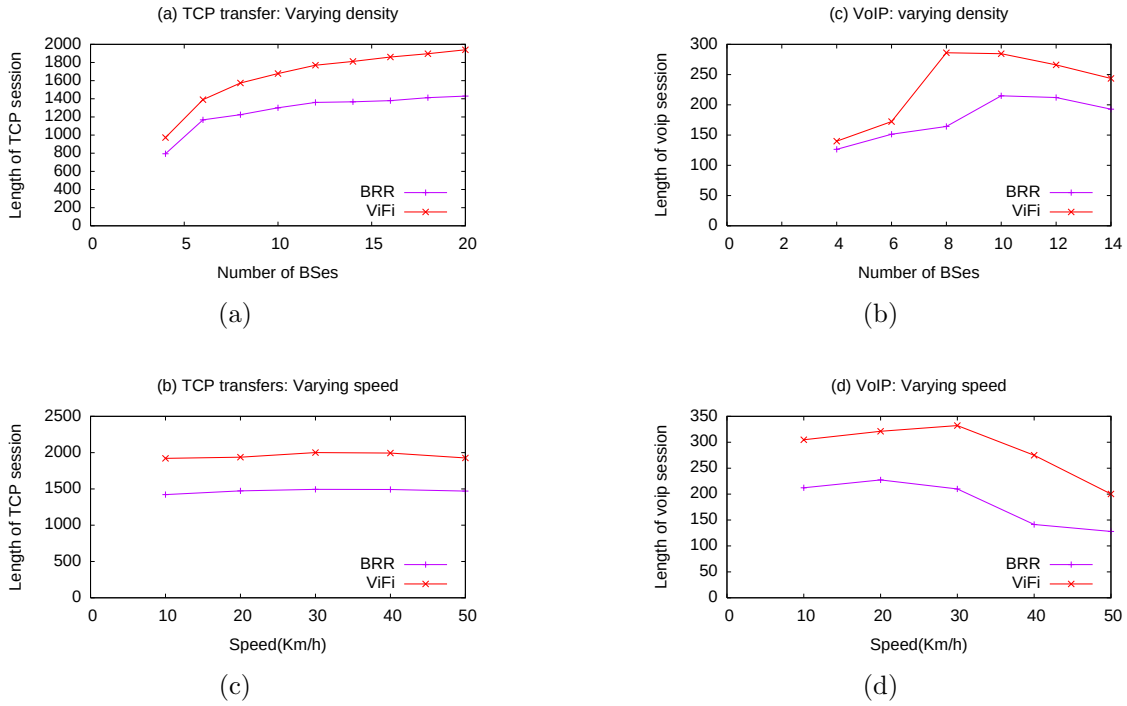


Figure 5.19. ViFi: Comparisons between BRR and ViFi using the QualNet simulated environment under varying speed and density.

5.4.7 Limitations

Finally, we tested the relaying mechanism of ViFi in a range of simulated conditions to understand where it might perform poorly. We find two such conditions. First, when the number of auxiliary BSes is high (e.g., greater than 15). Second, all auxiliary BSes are equi-distant from both the source and the destination. In both conditions, while the average number of relays per packet is one (Eq. 5.1), the variance in the number of relays per packet increases, resulting in higher false positives and negatives. Neither of these situations arise in our testbed environments. To make ViFi robust in environments where they might, it can be extended such that the number of auxiliary BSes is limited or the symmetry between them is broken. These extensions are subject of future work.

5.5 ViFi conclusions

In this chapter, I establish the thesis statement (Section 1.1) in the context of mesh networks. Specifically, I show that leveraging opportunistic overhearing reduces disruptions and enables highly interactive applications in vehicular-mesh networks. Leveraging opportunistic overhearing by neighboring nodes introduces a resource management question — Of the multiple APs that overhear a packet, which AP should forward the overheard packet to the next hop? ViFi solves the resource management problem using a utility-driven sender prioritization protocol.

I answer the research questions raised in Section 1.2 with respect to the mesh environment using measurement, protocol design, and testbed evaluation. Using a measurement study on the VanLAN testbed, we characterize disruptions in vehicular-mesh networks. We find that with current WiFi handoff methods, clients experience frequent disruptions in connectivity even when they may be close to WiFi AP. Further, we find that such disruptions do not significantly affect performance of bulk-transfer applications, but severely affect performance of interactive applications. Our measurement study shows that leveraging opportunistic overhearing can significantly reduce disruptions. It overcomes the limitations of using a single AP for reception because of independence of packet losses across APs.

Next, we design the ViFi protocol to exploit opportunistic overhearing. The challenge in designing ViFi is in coordinating among APs that opportunistically overhear packets. If every AP that overhears the packet relays the packet to the next hop, there is a significant waste of resources. IViFi addresses this challenge using a utility-based prioritization protocol. ViFi nodes run the prioritization protocol to decide the relaying probability with which each neighbor should forward the overheard packet. The prioritization is decentralized, where each node computes its own relaying probability without need for per-packet coordination. The prioritization problem is formulated to minimize the number of wasted transmissions, while reducing packet losses.

We implement and deploy ViFi on the VanLAN testbed for over two months. We evaluate ViFi using our deployed prototype and simulations driven by traces from our Dome-Mesh testbed. We study the performance of ViFi for two commonly used interactive applications: VoIP and short TCP transfers that are typical in Web browsing. Our evaluation using deployment and trace-driven simulations shows that the link-layer performance of ViFi comes close to an ideal oracular protocol that leverages multiple APs. We show that the link layer performance improvement translates to better application performance. In our deployment, ViFi doubles the number of successful short TCP transfers and doubles the length of disruption-free VoIP sessions compared to an existing WiFi-style handoff protocol. Trace-driven simulations on our Dome-Mesh testbed corroborate our findings.

CHAPTER 6

WIFFLER: AUGMENTING 3G CONNECTIVITY

Cellular access is the most preferred means for mobile connectivity among urban users today. There is widespread deployment of cellular infrastructure by providers that offer near-perfect connectivity to mobile users. As a result, millions of users are subscribing for cellular service, with an expected subscription of 1 billion by 2011 [12].

However, cellular spectrum is limited, and the popularity of cellular networks is creating immense pressure on the spectrum. Subscribers, especially in big cities, are experiencing deteriorating 3G quality because the network cannot cope with the high demand [115]. In response to this pressure, wireless providers are attempting to reduce usage by such as imposing a limit of 5GB per month [121] and "educating" their users on "responsible" access [106]. Such methods are unlikely to be sufficient in the long run.

In this chapter, I present the Wiffler protocol that answers the following question: *Can cellular spectrum be augmented using WiFi connectivity for vehicular users?*. WiFi networks operate in the unlicensed spectrum and a WiFi AP only provides coverage to only a small area of around 100-200 meters. In contrast, 3G cellular towers have a much larger range of 1 Km and can provide connectivity to a larger number of users, and therefore have more demand on their spectrum. Augmenting the long range 3G network with short range and cheap WiFi connectivity can reduce the spectrum pressure on 3G.

As a first step, we conduct a joint measurement study of cellular and WiFi networks in 3 cities, to understand the disruption and performance characteristics of

the two networks. Our study shows that 3G is only available 87% of the time, but combining 3G and WiFi can reduce disruptions/unavailability in 3G networks by 50%. However, in terms of reducing 3G spectrum pressure, our study suggests that straightforward methods of combining 3G and WiFi can reduce 3G load by at most 11%, and even that will come at the expense of poor application performance.

We then design Wiffler, to overcome these availability and performance challenges. Its two key ideas are *leveraging delay tolerance* and *fast switching to 3G.* We observe that many applications, such as email or file transfer, can afford to delay data transfers without significantly hurting user experience. Wiffler leverages this observation: Using a simple method to predict future WiFi throughput, it delays data transfer to wait for WiFi connectivity, but only if 3G savings are expected. Additionally, so that the performance of delay and loss sensitive applications is not hurt, Wiffler quickly switches to 3G if WiFi is unable to transmit the packet within a time window.

The rest of this chapter is organized as follows: In Section 6.1, I describe related work and in Sections 6.2, I present results from our measurement study. Section 6.3 describes the Wiffler protocol. In Sections 6.4 and 6.5, I present Wiffler evaluations using deployment experiments and trace-driven experiments, respectively. Section 6.6 concludes this chapter.

6.1 Related Work

In Wiffler, we characterize vehicular connectivity using a joint 3G/WiFi measurement study, and predict future connectivity to augment 3G with WiFi. Below, I describe research related to characterizing connectivity from moving vehicles, predicting future connectivity, and augmenting a network interface.

Characterizing vehicular connectivity. Several studies have characterized WiFi and 3G connectivity in isolation for vehicular settings. For WiFi, the CarTel study quantifies the frequency of AP encounters and the throughput that can

be achieved using open APs [61]. Various researchers have since studied link-layer characteristics [76, 22], TCP throughput [59], as well as the performance of specific applications (e.g., web search [19]) and handoff policies [54]. Similarly, for 3G, several recent works have studied characteristics such as signal strength, loss rate, latency, and TCP throughput [75, 85].

In contrast, we conduct a joint characterization that enables a head-to-head comparison of 3G and WiFi. For any one technology, our results are qualitatively consistent with the studies above, but our joint characterization is crucial to understanding and leveraging their combined power.

Predicting future connectivity. *Breadcrumbs* predicts future WiFi connectivity based on a model of the environment [84]. Similarly, Deshpande et al. [46] use WiFi prediction to improve mobile access. Both of these schemes rely on the existence of an AP location database that provides the WiFi capacity of APs in different locations. In addition, breadcrumbs uses mobility prediction as part of its model, which means that clients must store and estimate the transition probabilities between different locations.

In contrast, our model does not require an external database, and predicts based only on a short meeting history. In our evaluation, we compare the performance of Wiffler with a Breadcrumbs-like prediction model.

Using multiple interfaces. In Section 2.3.4, I describe several related projects that propose mobile systems that augment one network using a second available network. The goal of these previous works is to improve performance or energy efficiency. In contrast our primary goal is to offload as many bytes on the second network as possible, while satisfying an application-specific performance requirement. Thus, instead of responding purely to current conditions, we also base decisions on predictions of future conditions.

6.2 Measurement

We conduct a joint study of 3G and WiFi network characteristics. We seek to answer the following questions: *(i)* What is the availability of 3G and WiFi networks as seen by a vehicular user? *(ii)* What are the performance characteristics of the two networks?

6.2.1 Testbeds and methodology

We conducted measurements in three geographically separate, outdoor testbeds that include effects present in real vehicular settings, such as noise, fading, interference, occlusions, and traffic patterns. We refer to the three testbeds as *Amherst*, *Seattle*, and *Sfo*.

Amherst is located in Amherst, MA and uses the Dome-3G testbed described in Section 2.2.. The vehicles are equipped with a 3G data modem, in addition to the other components available on the Dome vehicles. The 3G modem has HSDPA-based service via AT&T. We collected more than 3000 hours of measurement data from *Amherst* over 12 days. Cumulatively, over 500 GB of data was transfered.

The vehicles in *Amherst* go through areas of sparse connectivity for the most part, but also are within range of 1.5 sq mile area of a mesh environment. When vehicles are in the mesh environment, they connect to the mesh APs. In the remaining areas, the vehicles connect to open, reachable APs. In our data, over 70% of the connections are through non-mesh APs [101].

The software on the vehicles includes two main programs. The first program scans the WiFi and 3G channels simultaneously and obtains an IP address whenever a connection is available. Once a connection is established on any interface, the second program sends and receives data to a server. Both the server and the vehicle log the characteristics of the duplex data transfer on the WiFi and the 3G interfaces. More than 55% of the APs that vehicles in *Amherst* encountered are closed. Nevertheless,

the vehicles were able to successfully exchange data with more than 100 unique open WiFi APs each day.

Seattle is located in the metropolitan region of Seattle, WA. It consist of one vehicle that is equipped with the same hardware and software as the vehicles in *Amherst*. Measurements in *Seattle* include large portions of highway driving and we present results for data collected over 6 days. From the single vehicle, about 5GB of data were sent and received over the course of the experiment.

Sfo is located in a metropolitan region of San Francisco, CA. We presents results from 3 days of data, gathered from one vehicle using the same setup as the oher two testbeds.

The vehicles in both *Seattle* and *Sfo* exchange data with open APs that we did not deploy. Unlike *Amherst*, in which the buses scheduled routes, *Seattle* and *Sfo* data were collected using unscheduled driving patterns that did not follow a regular path.

All of our measurements and performed on the AT&T network and as a result, our conclusions are specific to the AT&T network.

6.2.2 Availability of 3G and WiFi

To measure availability, the vehicle and the server periodically send data to each other over UDP. Availability is measured over 1-second intervals. In each interval, an interface (WiFi or 3G) is considered available if at least one packet was received in the interval. Availability is defined as the number of available 1-second intervals divided by the total number of intervals.

Figure 6.1 shows that availability in each testbed. We see that in *Amherst*, 3G and WiFi are available 90% and 12% of the time, respectively. Interestingly, the percentage of time neither 3G or WiFi is available is only 5%. In other words, combining 3G and WiFi can reduce 3G unavailability from 10% to 5%, a 50% reduction. The combination of WiFi and 3G reduces unavailability significantly because of a

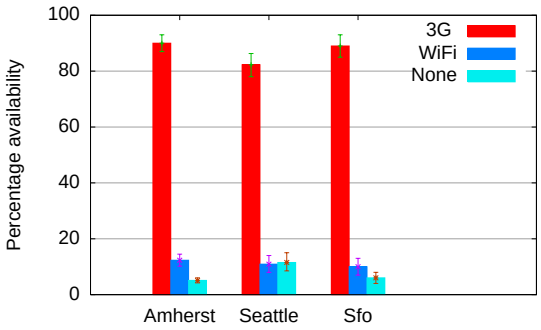


Figure 6.1. Wiffler: 3G and WiFi availability on the three testbeds.

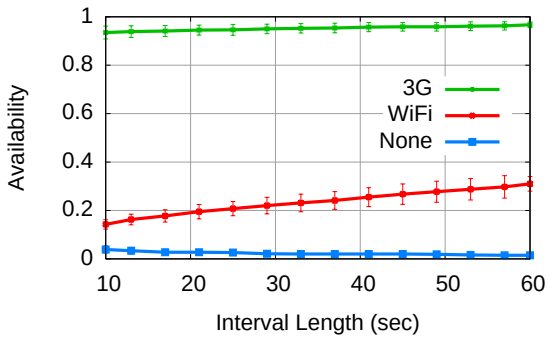


Figure 6.2. Wiffler: 3G and WiFi availability in *Amherst* at longer time intervals.

negative correlation between the availability of 3G and WiFi. Out of the 12% WiFi availability, 5% of the WiFi availability is when 3G is not available. If WiFi and 3G availability were completely independent, the overall unavailability even when both 3G and WiFi are combined would be $(1 - 0.90)(1 - 0.12) = 9\%$.

Figure 6.1 shows that in *Seattle*, 3G availability is only 82%, and the WiFi availability is 10%. When 3G and WiFi are both considered, network unavailability is 11%. Again, if only the 3G interface is used, the unavailability would be roughly 16%. Similarly, in *Sfo*, 3G availability is 89% leading to an unavailability of 11%. But when combined with WiFi, the total unavailability reduces to 5%. Figure 6.1 shows that the negative correlation between 3G and WiFi is not specific to *Amherst*, but can also be observed in *Seattle* and *Sfo*.

In summary, in all three testbeds, network unavailability is reduced by over 50% by combining WiFi and 3G compared to using 3G alone. We were not able to uncover the reason for the negative correlation observation.

6.2.2.1 Availability over longer intervals

For less-demanding applications, such as email or file transfer, intervals longer than 1 second are more appropriate for measuring availability. Figure 6.2 shows the availability of 3G and WiFi over longer time intervals, from 10 to 60 seconds. The

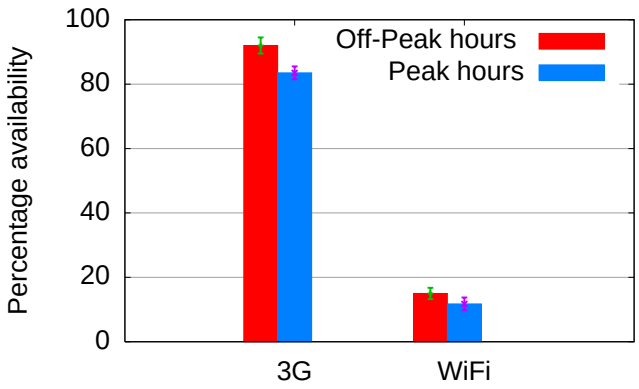


Figure 6.3. Wiffler: Comparing 3G and WiFi availability during peak and off-peak hours in *Amherst*.

results are based on *Amherst* measurements. 3G is available (i.e., at least one packet is received in an interval) close to 98% of the time with 60-second intervals. The availability of WiFi also increases with 60-second intervals, to 30%. We observed qualitatively similar effects in *Seattle* and *Sfo* (not shown in figure).

6.2.2.2 Availability in peak vs. off-peak hours

We study availability of 3G and WiFi with respect to time of day, in particular, during *peak* and *off-peak* hours. We define peak hours to be from 8.00 AM to 9.00 PM and off-peak hours to be from 9.00 PM to 8.00 AM.

The vehicles in *Amherst* operate throughout the day but operate in fewer locations during off-peak hours. For a fair comparison, we only consider locations where the vehicle operates during both the peak and off-peak hours. We perform the experiment using days when the vehicles were operational during both peak and off-peak hours for at least 2 hours. The availability is computed as an average availability during 2-hour intervals.

Figure 6.3 shows the 3G and WiFi availability during peak and off-peak hours in *Amherst*. The results are computed over 4 days and the error bars show the 95% confidence interval. 3G availability during off-peak hours in *Amherst* is 9%

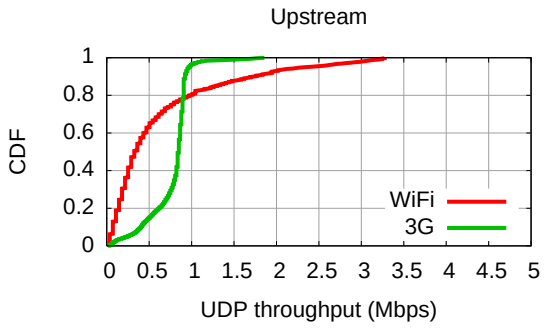


Figure 6.4. Wiffler: Upstream UDP throughput in *Amherst*.

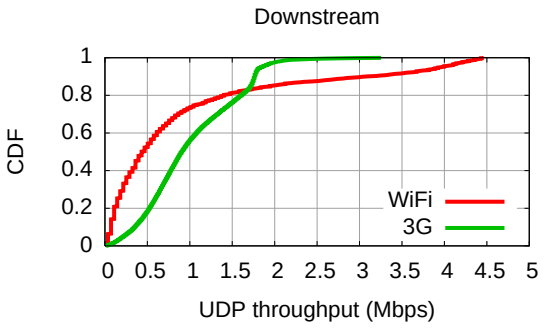


Figure 6.5. Wiffler: Downstream UDP throughput in *Amherst*

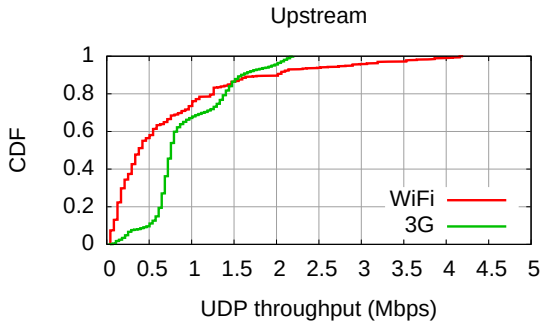


Figure 6.6. Wiffler: Upstream UDP throughput in *Seattle*.

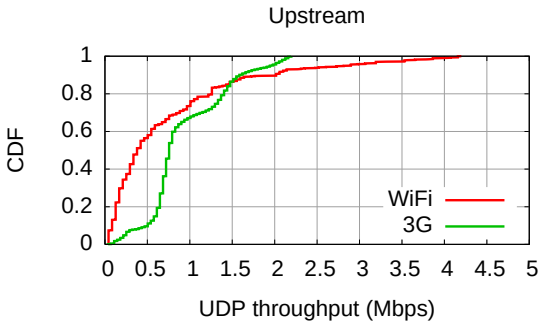


Figure 6.7. Wiffler: Downstream UDP throughput in *Seattle*.

greater than the availability during peak hours. WiFi availability during peak hours is lower by 4%. We find that similar trends hold in *Seattle* as well. The difference in 3G availability between peak and off-peak hours is 6%, and the difference in WiFi availability is less than 3%.

The results indicate that the performance of 3G and WiFi networks suffers during periods of high spectrum use (i.e., during peak hours). But our experiments do not provide stronger evidence to show causality between availability and spectrum use.

6.2.3 Performance of WiFi and 3G

We use three measures—UDP throughput, TCP throughput, and loss rate—to characterize the performance of 3G and WiFi networks. To measure the upstream

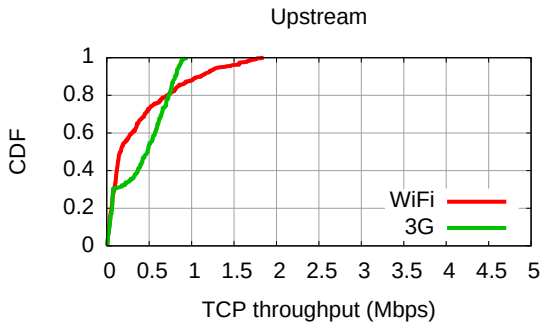


Figure 6.8. Wiffler: Upstream TCP throughput in *Amherst*.

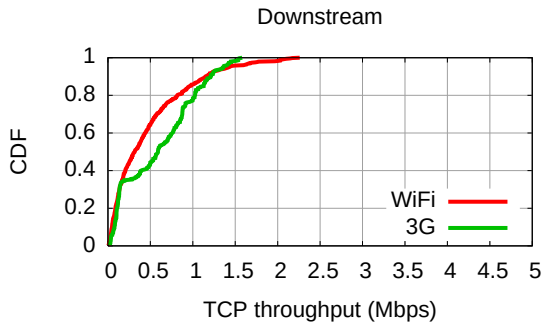


Figure 6.9. Wiffler: Downstream TCP throughput in *Amherst*.

and downstream UDP throughput, the vehicle and server each send 10 back-to-back 1500-byte packets every 20 ms. We measure UDP throughput in all three testbeds. To measure loss rate, the vehicle and server each send a 20-byte packet every 100 ms. To measure TCP throughput, the vehicle and the server each create a TCP connection and send 100KB data to each other repeatedly. At the end of a 100KB transfer, the TCP connection is closed and a new connection is created. We measure loss rate and TCP throughput only in *Amherst*. All performance results are based on at least 3 days of measurement data.

6.2.3.1 UDP throughput

Figures 6.4 and 6.5 show the CDF of the 3G and WiFi UDP throughput in the upstream and downstream directions. All CDFs are generated using measurements over 1-second intervals. They include points only for intervals with non-zero throughput (i.e., non-zero availability). The 3G lines thus have almost 10 times as many points as the WiFi lines.

In the upstream direction, 3G and WiFi achieve a median UDP throughput of 850 Kbps and 400 Kbps respectively in *Amherst*. In the downstream direction, the median 3G throughput is again about twice that of WiFi. For example, in *Amherst*, we observe a median 3G throughput of 1Mbps and a median WiFi throughput

of 500Kbps. The median upstream and downstream UDP throughput is similar in *Seattle*, as shown in Figures 6.6 and 6.7. In both testbeds the median WiFi UDP throughput is about half of the median 3G throughput, but the top 20th percentile of WiFi outperforms 3G.

6.2.3.2 TCP throughput

Figures 6.8 and 6.9 show the upstream and downstream TCP throughput of 3G and WiFi in *Amherst*. In the upstream direction, the median TCP throughput of 3G and WiFi are 500 Kbps and 200 Kbps, respectively. In the downstream direction, the median TCP throughput of 3G and WiFi are 600 Kbps and 280 Kbps, respectively. Thus, the median TCP throughput is only about half of the median UDP throughput for both the 3G and WiFi networks. However the relative TCP performance of 3G versus WiFi is similar to the relative UDP performance.

Taken together with the UDP measurements, the results above suggest that the throughput performance of WiFi in mobile outdoor environment is poorer than 3G. The result points to an important difference between stationary and mobile environments. In typical stationary settings, WiFi throughput is significantly higher than 3G throughput.

6.2.3.3 Loss rate

Figure. 6.10 shows the loss rates over 1-second intervals for 3G and WiFi in *Amherst*. We see that 3G loses no packets in 93% of the intervals. WiFi has no packet loss in 78% of the intervals but loses all packets in 12% of the intervals. In other words, in 90% of the intervals WiFi delivers no packet or delivers all of them. This behavior is consistent with prior studies that have shown that WiFi losses are bursty in both indoor and vehicular settings [104, 22] and losses are bi-modal. These bursty losses make WiFi offloading challenging, especially for applications with strict QoS requirements such as VoIP or video conferencing.

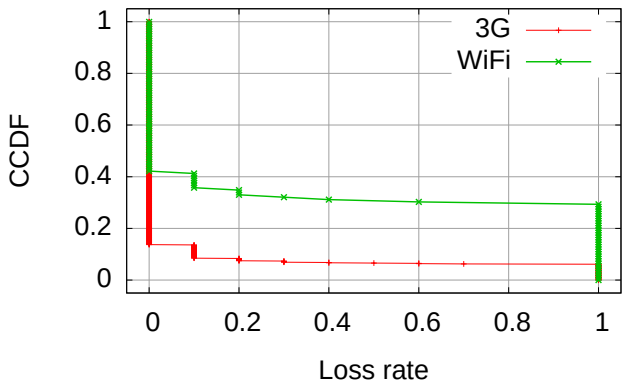


Figure 6.10. Wiffler: 3G and WiFi loss rate in *Amherst*.

6.2.3.4 Spatial variations in performance

Using data collected in *Amherst*, we study the geographical distribution of 3G and WiFi performance. Our goal is to characterize the locations where WiFi can augment 3G connectivity. We divide the geographical area into grids and compute the total data transfered over the 3G and WiFi per unit time spent in the grid, averaged over a day.

Figure 6.11 compares the performance of 3G and WiFi at different grid locations. In all, there were 120 grid locations in which packets were received on either WiFi or 3G at least once. In 47% of the grid locations, the total data sent on WiFi is insignificant compared to the data sent over 3G. In the remaining 53% of the grid locations, at least 20% of the 3G data could be shifted to WiFi. In 9% of the grid locations, equal or more data was sent over WiFi than 3G, i.e., all 3G traffic could be offloaded to WiFi.

6.2.4 Measurement summary

In summary, the measurement study shows that

- The availability of WiFi is an order of magnitude poorer than 3G.

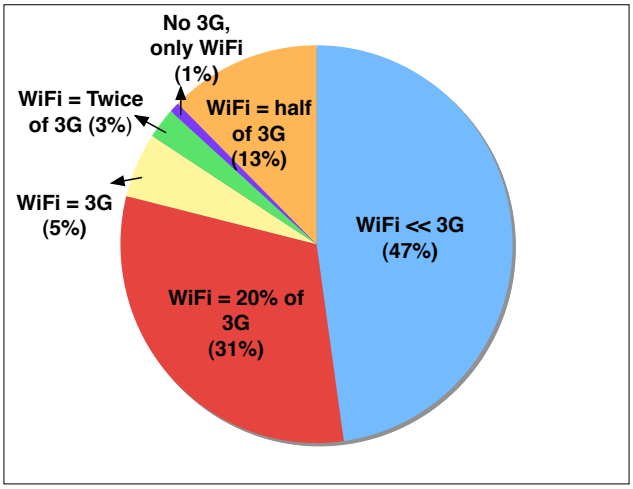


Figure 6.11. Wiffler: The spatial distribution of 3G and WiFi performance in *Amherst*. The *Amherst* testbed was divided into grids of size is 0.5 miles × 0.5 miles.

- The WiFi loss rate performance is also poorer compared to 3G. Therefore, leveraging WiFi to augment 3G may incur performance penalties. WiFi throughput is also much lower than 3G throughput.

6.3 Wiffler Protocol

The goal of Wiffler is to reduce 3G usage by leveraging WiFi connectivity when available, but to do so without affecting application performance.

The simplest policy for using WiFi is to send data on the WiFi network when available and switch to the 3G network when WiFi is unavailable. Results from our measurement study show, however, that this policy does not work well in practice because of two key challenges. First, the average availability of WiFi can be low—11% in our measurements—which severely limits the fraction of data that can be offloaded to WiFi. Second, WiFi loss rate is higher than 3G. For applications that are sensitive to losses, such as VoIP, using WiFi irrespective of its loss characteristics will degrade application quality.

Wiffler uses two ideas to address these two challenges: *leveraging delay tolerance* and *fast switching to 3G*. The key insight in the former is that delay tolerance allows applications to trade-off completion time for 3G usage. A user may be willing to tolerate a few seconds delay to send their email or complete a file transfer if it reduces 3G usage. Wiffler delays data transfers to reduce 3G usage, but only delays a transfer if the added delay results in 3G savings and is within the application's delay tolerance.

For applications with strict quality of service requirements, such as VoIP and video, Wiffler uses the fast switching mechanism. When using WiFi, it promptly switches packets to 3G if WiFi cannot deliver them within a certain time period.

6.3.1 Wiffler API

Wiffler takes as input application data, which is characterized using S, the size of the transfer, D, the delay tolerance and an application-specified QoS metric. Based on these characteristics and those of the operating environment, it decides how to distribute the data across 3G and WiFi.

6.3.2 Leveraging delay tolerance

Wiffler estimates future WiFi throughput and delays transfers only if the estimate indicates that delaying transfer will reduce 3G usage. Wiffler's's prediction method is described in Section 6.3.4. For now, assume that we have a predictor that yields a (possibly erroneous) estimate of WiFi capacity from the current time until a future time.

Wiffler uses the predictor to estimate offload capability of the WiFi network until the delay tolerance threshold. The decision to either wait for a potential WiFi offload opportunity or to send immediately on 3G is made based on the predicted WiFi capacity and the application workload. For example, one possible strategy is to wait for WiFi only if all of the application data can be transfered over WiFi before the delay tolerance threshold. Since the estimate can be wrong, an alternative, more

conservative strategy is to wait for WiFi only if the predictor estimates that twice the application data can be transfered over WiFi before the delay tolerance threshold. The completion time versus 3G savings trade-off for these two strategies is clearly different.

To capture this trade-off, we introduce a tuning parameter called the *conservative quotient*. The conservative quotient, c, is a number between 0 and ∞. For a given value of c, the Wiffler offloading protocol is shown in Figure 6.12. The protocol considers the total data S that needs to be transfered within the earliest delay tolerance threshold, and the total data the node can transfer on WiFi, W. The next two steps are done in parallel. If WiFi is available, we use it immediately to transfer data. 3G connectivity is used only if we estimate that $W \leq S \cdot c$.

If $c < 1$, Wiffler will wait for a WiFi offload opportunity even if only a fraction c of the total application data can be transfered on WiFi in expectation. Therefore, this strategy will offload more data on WiFi at the expense of completion time. On the other hand, if $c > 1$, Wiffler waits for WiFi only if the WiFi capacity is substantially greater than the load. Therefore, the completion time of the strategy is likely to be lower, but it also has a lower offload potential. Unless stated otherwise, we set $c = 1$ in our experiments.

The conservative quotient can be set not only by the system or the application but also by the 3G provider. For example, during peak times when 3G spectrum pressure is high, the provider may decide to offload more data on WiFi at the expense of application latency and set c to a small value. But during the off-peak times, c can be increased to improve application latency.

6.3.3 Fast switching to 3G

lin as video streaming and VoIP are sensitive to even small delays and losses. Because of a greater chance of loss, using WiFi to transfer such data can hurt appli-

D: earliest delay tolerance threshold among queued transfers
S: size in bytes to be transferred by D
W: estimated WiFi transfer size

if (WiFi is available):

- send data on WiFi and update S

if ($W < S \cdot c$ and 3G is available):

- send data on 3G and update S

Figure 6.12. Wiffler: Prediction-based offloading protocol.

cation performance. Thus, if WiFi is losing or delaying packets, we should send them on 3G as soon as possible.

Wiffler uses low-level, link-layer information to enable fast switching to 3G in the face of poor WiFi conditions. We added a signaling mechanism in the mobile node's driver that signals the application when the wireless card receives a link-layer acknowledgement. The signal contains the identity of the acknowledged packet. The application matches the acknowledgement with its outstanding packets. If the application does not receive a link-layer acknowledgement for a packet before a delay threshold, it sends the packet on the 3G interface. We set the delay threshold to 50 ms.

Link layer information is needed because the WiFi NIC frequently takes a long time to complete retransmission attempts. For instance, the driver that we use in our testbed (*Madwifi*) retries packets 11 times, which even if we ignore medium access delays takes more than 120 milliseconds with the default 802.11b specification. This delay can affect performance of applications such as VoIP.

Our fast switching mechanism is simple: it sends the packet on 3G if the WiFi link-layer fails to deliver the packet within a delay threshold. The motivation for this protocol is that waiting for WiFi link-layer retransmissions incurs delays. In addition, when a packet is lost, there is a high chance that the retransmission will

fail, since losses are bursty in the vehicular environment [22, 104]. Thus, it is better to send time-sensitive packets on 3G rather than waiting for likely more failures on WiFi. Choosing the delay threshold involves a trade-off between better application performance and 3G load. In Section 6.5, we analyze this trade-off in detail.

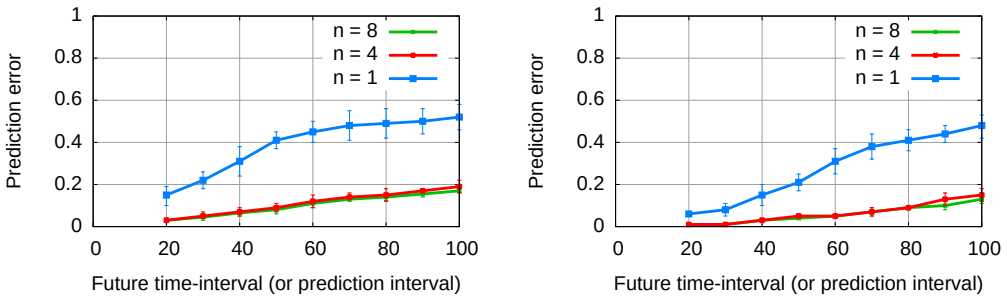
6.3.4 WiFi throughput prediction

We predict WiFi offload capacity based on an estimate of the average throughput offered by an AP and a prediction of the number of APs that will be encountered until a given future time interval.

Our prediction of AP encounters is based on the observation that AP meetings occur in bursts. That is, if the mobile node meets APs frequently (e.g., because it is in a dense urban area with many APs), then the node is likely to meet the next AP within a short time interval. Similarly, if the mobile node has not met an AP for a long period of time (e.g., because it is on a highway), then the node is unlikely to meet an AP within a short time interval. An analysis of our measurement data shows that AP meetings in reality indeed have this property.

Based on this observation, we predict the number of AP encounters using a simple history-based predictor. The mobile node keeps track of the last n AP encounters and computes the average time between the encounters. Wiffler predicts the number of AP encounters until a future time using the average inter-meeting time of the past encounters. For example, if the average inter-meeting time of the past encounters is I seconds, then Wiffler predicts the number of AP encounters in the next T seconds to be $\frac{T}{I}$. Similarly, the average throughput is estimated based on the throughput observed by the vehicle at each AP encounter.

We study the accuracy of the AP encounter prediction using the traces we gathered from our testbeds. Figure 6.13 shows the AP prediction error for different values of n, the number of previous encounters used in the prediction. The prediction error is



(a) Amherst (b) Seattle

Figure 6.13. Wiffler: The relative average error between the number of APs predicted and the number of AP meetings observed in the measurement. Based on measurements collected from *Amherst* and *Seattle*. Vertical bars shows the 95% confidence interval around the mean.

presented for different future time-intervals (or prediction intervals). We compare the predicted number of encounters with the actual number of encounters over 10-second time periods, and present the average.

Figure 6.13(a) shows for the *Amherst* testbed that if the prediction is based on only one previous AP encounter (n=1), the prediction accuracy is low. The prediction error is close to 20% even for predicting AP encounters until a small future time-interval of 20 seconds. On the other hand, when prediction is based on the previous 4 or 8 AP encounters, the prediction error is less than 5% up to a future prediction time-interval of 50 seconds. The prediction error increases to 20% for a prediction time-interval of 100 seconds. Figure 6.13(b) shows that AP prediction yields high accuracy in the *Seattle* testbed as well, even though the vehicle did not follow preset routes, unlike the vehicles in the *Amherst* testbed.

Since in both testbeds the accuracy of prediction based on 8 previous encounters is similar to the prediction based on 4, our experiments use n=4.

Our simple prediction framework allows us to estimate the WiFi offload capacity with no pre-programmed knowledge about the environment. More complicated prediction models that use additional information about the environment exist in the

	Completion time	% offloaded to WiFi
Wiffler **offloading**	45 sec	30%

Table 6.1. Wiffler: Deployment results of prediction-based offloading.

literature. For example, if the AP locations are available *a priori*, then the WiFi offload capacity can be predicted by predicting user mobility, instead of AP prediction [84, 46]. In Section 6.5, we show that the marginal improvement obtained by using location information is small.

6.4 Deployment results

We implemented both Wiffler's's prediction-based offloading and fast switching and deployed it on the Dome-3G testbed. We evaluated the performance of Wiffler with respect to delay tolerant file transfer and highly interactive Voice Over IP.

6.4.1 Prediction-based offloading

This experiment uses a deployment of Wiffler on 20 nodes over a period of 2 days. Each node generates 5Mb (or 5000,000 bytes) of application data. The data is generated as a uniform process with a mean interval of 100 seconds. We set the delay tolerance threshold for data delivery to be 60 seconds. All data is destined to a known server that we control.

Table 6.1 shows the results. For 5Mb transfers and a deadline of 60 seconds, Wiffler reduces 3G usage by 30%, even though the WiFi availability is only 12% (Section 6.2.2).

6.4.2 Fast switching

We evaluate fast switching in the context of VoIP. We assume that the VoIP application uses the popular G.729 codec and generates 20-byte packets every 20 ms.

	% time voice quality good	% offloaded to WiFi
Fast switching	68%	34%
WiFi when available	42%	40%

Table 6.2. Wiffler: Deployment results for VoIP using fast switching.

We calculate VoIP quality by using the standard MOS metric that ranges between 1 (unacceptable) and 5 (best). To evaluate VoIP performance in a quickly changing environment, we estimate the MOS value for 3-second intervals and quantify the overall quality as the fraction of intervals where the MOS value is more than 3.0. 3-second is roughly the time it takes to enunciate a short English sentence. The methodology to estimate the MOS score is described in Chapter 5, Section 5.4.3.2.

Implementing fast switching in the downstream direction is challenging. It needs either support from the APs or detailed information at the proxy on current WiFi conditions. In this work, we implement fast switching only in the upstream direction. Our trace-driven simulations study the benefit of fast switching in the downstream direction as well.

Table 6.2 shows the results using one vehicle in our deployment that operated in an area with high WiFi availability. Fast switching maintains good voice quality for over 68% of the time and reduces 3G usage by 34%. Instead, if we used WiFi whenever available, without switching to 3G during periods of bad WiFi quality, voice quality is maintained only 42% of the time, even though the 3G savings marginally increases from 34% to 40%.

6.5 Trace-driven evaluation

We now present extensive trace-driven evaluation of Wiffler. We consider a range of different conditions as well as compare Wifflerto alternatives strategies for offloading data.

6.5.1 Evaluation of prediction-based offloading

To evaluate Wiffler's prediction-based offloading, we use the TCP throughput traces collected during our measurements. The traces provide information on how much data can be sent or received on 3G and WiFi during 1-second intervals. We show below that our trace-driven simulations yield results similar to those in our deployment.

We characterize offloading performance using two metrics: *(i)* the fraction of data sent over WiFi, which measures the reduction in 3G usage; and *(ii)* the average completion time.

We evaluate the performance of Wiffler and some alternate offloading strategies over three dimensions.

- *Workload:* We use realistic application workloads as well as synthetic workloads.
- *Location:* We use traces collected from *Seattle* and from *Amherst*. We also evaluate the performance of the protocols in areas with higher AP density.
- *Application conservativeness:* We use different conservative quotients and delay tolerance thresholds.

6.5.1.1 Alternative strategies

We compare Wiffler with several alternative strategies to offload data to WiFi. We consider three classes of alternative algorithms.

Algorithms without prediction: To understand the value of prediction, we evaluate two algorithms that do not use prediction. The *Impatient* algorithm uses a very simple policy: use 3G whenever WiFi is unavailable; else use WiFi. The *Patient* algorithm waits and sends data on WiFi until the delay tolerance threshold, and only

switches to 3G if all of the data are not sent on WiFi before the delay tolerance threshold. *Patient* and *Impatient* present the two extreme points in the design space.

Algorithms with prediction: To understand the accuracy versus complexity trade-off, we compare Wiffler's simple prediction scheme against a more sophisticated prediction model that we call *Adapted-Breadcrumbs*. This model is similar to the Breadcrumbs system [84]. At each location grid, the system learns the available WiFi bandwidth and the probability of the client moving to an adjacent grid. It forecasts WiFi transfer sizes by taking the weighted average of expected transfers at each future grid. We use grid sizes of 0.2 miles $\times$ 0.2 miles and the learning phase uses the previous day of data.

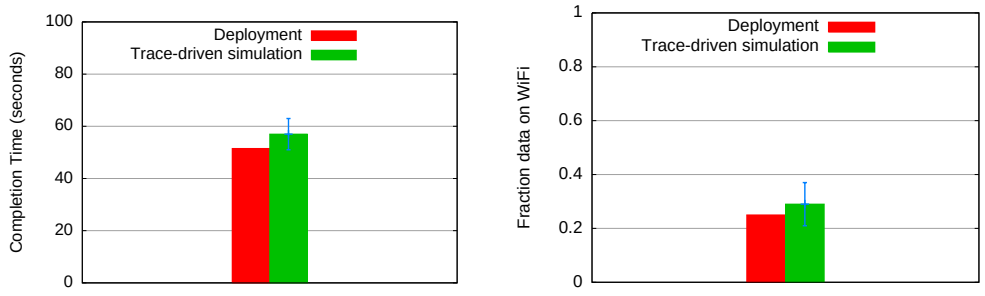
Algorithm with future knowledge: To quantify the remaining room for improvement, we also consider an (impractical) algorithm with perfect future knowledge that we call *Oracle*. *Oracle* knows the exact amount of data that can be transfered using WiFi within the delay tolerance threshold, and uses this knowledge to make a decision about when to use the 3G network. It minimizes 3G usage with the lowest achievable completion time.

6.5.1.2 Workload

We conduct our experiments using two workloads.

Realistic application workload: We obtained the workload from two corporate commuter buses that provide Internet access to the passengers. We sniffed the intra-bus WiFi network to capture packets that are sent and received by the riders. Based on the captured traces, we obtain distributions of connection sizes and inter-arrival times. We then generate realistic workloads based on the distributions. The average size of workload is 58 Kbps but it is highly bursty.

Synthetic workload: In order to experiment with a wider range of workload parameters, we generate a synthetic workload where a mobile node generates applica-



(a) *Completion time* (b) *Fraction data offloaded to WiFi*

Figure 6.14. Wiffler: Comparing the deployment versus simulation results.

tion data of size 5MB uniformly. The mean generation interval is set to 100 seconds. Similarly, a remote server generates transfers for each client at the same rate. Each experimental setting is run 10 times with different random seeds.

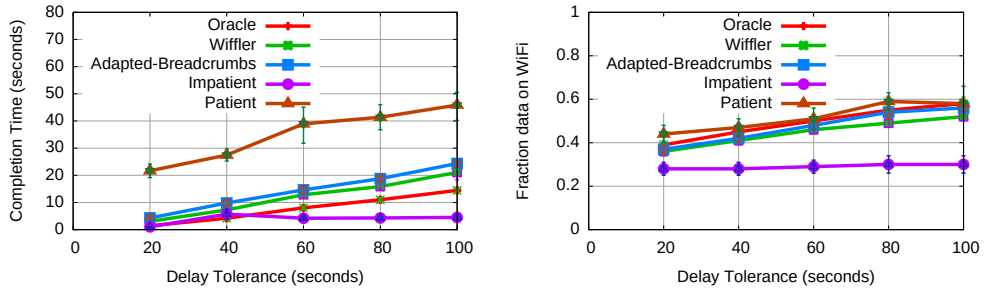
6.5.1.3 Validating trace-driven simulation

To validate the simulator, we collect throughput data during the deployment. During deployment periods when there are no application data to be sent or received, the vehicle transfers random data to the server both over WiFi and 3G and logs details of this transfer. As a result, the logs contain the throughput trace for the entire deployment duration. We conduct a trace-driven evaluation of Wiffler using this collected trace. We use the same packet generation parameters as the deployment. The simulation results are averaged over 10 runs with different seeds.

Figure 6.14 shows the performance of Wiffler observed in the deployment and in the simulator. Error bars show the 90% confidence interval. We see that the deployment results match well with the simulation results both in terms of completion time and percentage of data offloaded to WiFi.

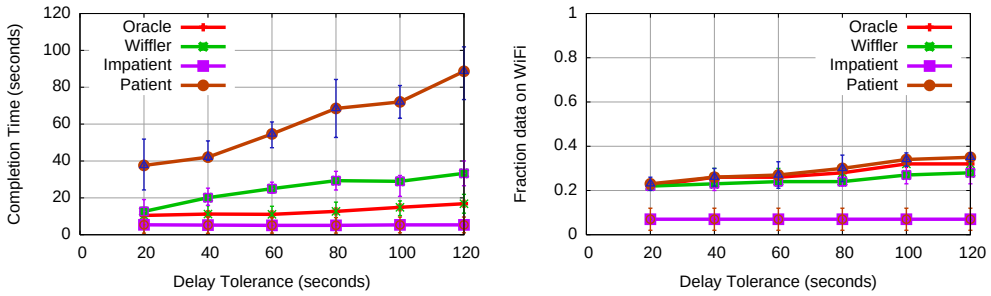
6.5.1.4 Realistic workload

Figure 6.15 shows the performance of the different offload algorithms for varying delay tolerance threshold in *Amherst*. Wiffler offloads a significant fraction of data



(a) *Completion time* (b) *Fraction data offloaded to WiFi*

Figure 6.15. Wiffler: Comparing offloading performance in *Amherst* with realistic application workload.

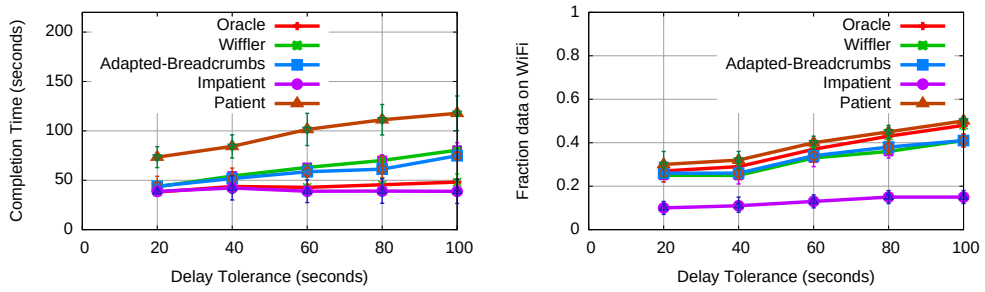


(a) *Completion time* (b) *Fraction data offloaded to WiFi*

Figure 6.16. Wiffler: Comparing offloading performance in *Seattle* with realistic application workload.

to WiFi. Figure 6.15(b) shows that if users are willing to wait 60 seconds, they can reduce 3G usage by 45%. The offload fraction increases as delay tolerance increases.

The *Patient* protocol reduces 3G usage by the most, because *Patient* sends data on WiFi opportunistically until the delay tolerance threshold. As a result, Figure 6.15(a) shows that the completion time using *Patient* is significantly higher than all the other protocols. In terms of completion time, the *Impatient* protocol performs the best since the protocol sends data on both 3G and WiFi and does not leverage delay tolerance. But as a result, *Impatient* reduces 3G usage by only 23% compared to the nearly 50% reduction achieved by other protocols, for a delay tolerance of 100 seconds.



(a) *Completion time* (b) *Fraction data offloaded to WiFi*

Figure 6.17. Wiffler: Comparing offloading performance in *Amherst* with synthetic workload.

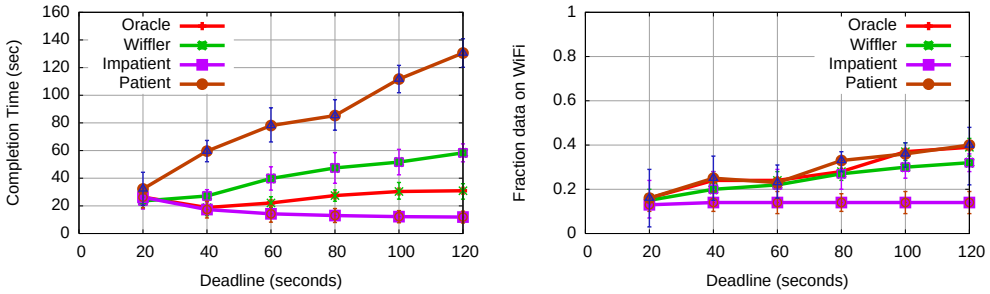
Oracle, with complete future knowledge achieves the optimal balance between reducing 3G usage and decreasing completion time. Wiffler performs within 5% of both *Oracle* and *Patient* in terms of 3G savings, and is within 7 seconds of *Oracle* with respect to completion time. In contrast, the *Patient* scheme that uses no prediction has a completion time that is 25 seconds more than *Oracle* on average.

Figures 6.15 shows that *Adapted-Breadcrumbs* performs similar to Wiffler both in terms of completion time and 3G savings even though *Adapted-Breadcrumbs* uses a more sophisticated prediction algorithm that learns WiFi performance in each location.

Figure 6.16 shows the performance of the offload protocols in *Seattle*. Because we did not measure TCP throughput in *Seattle*, we use UDP throughput for this experiment. As in *Amherst*, Wiffler provides about the same amount of 3G savings as *Oracle*. The completion time of Wiffler is within 10 seconds of *Oracle*.

6.5.1.5 Synthetic workload

We repeated the experiments above for a synthetic workload of 5Mb file transfers, to understand the performance of the different protocols with larger transfer sizes. Figure 6.17 shows the performance results for *Amherst*. We observe that less than



(a) *Completion time* (b) *Fraction data offloaded to WiFi*

Figure 6.18. Wiffler: Comparing offloading performance in *Seattle* with synthetic workload.

22% of data is offloaded to WiFi for small delay tolerance threshold. But for a delay tolerance of 100 seconds, Wiffler offloads 40% of data over WiFi.

Not surprisingly, Figure 6.17(a) shows that the completion time for the synthetic workload is higher than the completion time for the realistic workload, because of the larger data sizes. The completion time of *Patient* is nearly 75 seconds more than *Oracle*. Note that the average data size in the realistic application workload is 86Kbps compared to 5Mb in the case of synthetic workload. With larger data transfers, it is more likely that all of the data cannot be delivered using WiFi. This is both because of the lower throughput on WiFi and lower availability. As a result, in *Patient*, most transfers are completed only after the delay tolerance threshold, significantly inflating its completion time. Similarly, the difference in completion time between Wiffler and *Oracle* is about 35 seconds compared to only 5 seconds with the realistic workload that had smaller data transfer sizes (Figure 6.15(a)). Figure 6.18 shows that the results in *Seattle* are similar to those in *Amherst*.

6.5.1.6 Impact of AP density

3G cell towers are carefully placed to achieve near-complete coverage, but WiFi AP placement tends to be organic. In *Amherst*, certain areas have high AP density, but other areas have moderate to low AP density. As AP density is high typically

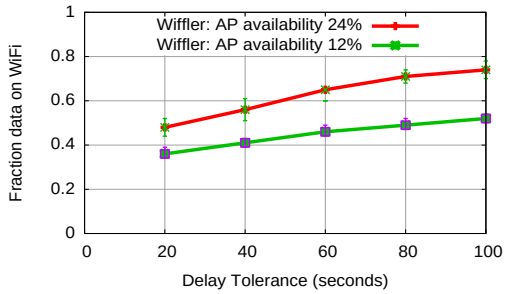


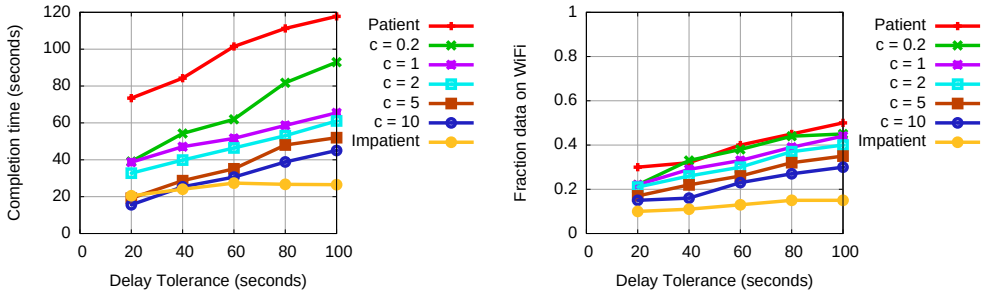
Figure 6.19. Wiffler: Comparing the fraction of data offloaded to WiFi under different AP availability conditions in *Amherst* with realistic workload.

in crowded downtown areas, where augmenting 3G capacity with WiFi is especially useful, we created a second data set. This *filtered* data set includes only measurements from a 15 sq. mile area with a high WiFi density. The availability of WiFi in this filtered data set is 24%, compared to 12% in the entire data.

Figures 6.19 show the performance of Wiffler in the total and the filtered data. In this experiment we used the realistic application workload. In areas with greater WiFi availability, 3G usage is reduced by 75% for a delay tolerance threshold of 100 seconds compared to a 50% reduction in 3G usage in regions with lower WiFi availability. The figure shows that even though the difference in WiFi availability in only 12%, the corresponding increasing in 3G savings is much higher. However, this is true only for large delay tolerance thresholds. For a lower threshold of 20 seconds, the difference in 3G savings between the two areas is only 9%.

6.5.1.7 Impact of conservative quotient

Wiffler uses prediction to trade-off completion time and 3G savings. As a result, the performance of Wiffler lies in between *Patient* and *Impatient*, the two extreme offloading strategies. In Section 6.3.2 we described an additional parameter called the *conservative quotient* c that allows Wiffler to achieve different trade-offs between completion time and 3G savings. Recall that Wiffler waits for WiFi only if the predicted WiFi capacity is c times the workload size.



(a) *Completion time* (b) *Fraction data offloaded to WiFi*

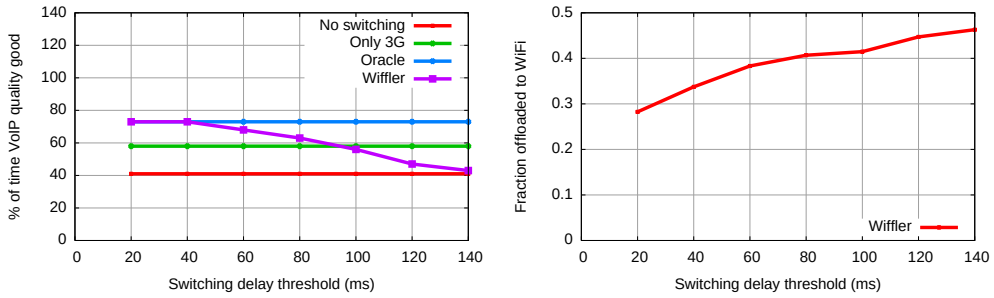
Figure 6.20. Wiffler: Trade-off between application latency time and 3G usage, in *Amherst* with synthetic workload.

Figures 6.20 shows the completion time and 3G savings for different values of c from 0.2 to 10. As the value of c increases, Wiffler starts sending data on the 3G interface much earlier instead of waiting for WiFi, which lowers completion time. On the other hand, the total data offloaded to WiFi when c=10 is significantly lower. When c=0.2, the total data offloaded to WiFi is 40% for a 100-second delay tolerance and the performance is close to the *Patient* protocol. On the other hand, the strategy has a poor completion time. The *conservative quotient* is thus an additional parameter that can be tuned to achieve different trade-offs. We find that c=1 offers a good trade-off between completion time and 3G savings.

6.5.2 Evaluation of fast switching

As with deployment experiments, we evaluate Wiffler's's fast switching in the context of VoIP. We use two performance metrics: *(i)* fraction of time that the VoIP quality is good; and *(ii)* fraction of data offloaded to WiFi. As before, we define VoIP quality to be good if the mean opinion score (MOS) values greater than 3.0 for a 3-second interval.

A goal of the evaluation is to understand the trade-off between VoIP quality and 3G savings for different values of switching delay threshold. A higher switching delay threshold increases 3G savings because there is a greater probability that the packet



(a) *VoIP performance* (b) *Fraction data offloaded to WiFi*

Figure 6.21. Wiffler: The performance of VoIP for varying switching time.

will be delivered using WiFi instead of 3G. However, the VoIP quality may be affected. Similarly, a lower switching delay threshold improves VoIP quality but may reduce 3G savings.

We compare the performance of fast switching in Wiffler to three other strategies. First, the *Only-3G* strategy supports VoIP using only 3G; the WiFi interface is never used. Second, the *No-switching* strategy does not switch away from WiFi as long as it is available. Finally, the *Oracle*strategy knows ahead of time if the packet will be lost on WiFi and, only in those cases, opts to send it on 3G. This strategy is impractical and serves as an upper bound on the performance.

For this trace-driven evaluation, we collected traces on *Amherst* by instrumenting one vehicle to send 20 byte packets every 20 ms to a server over both WiFi and 3G. Unlike the implementation, packets are sent both in the upstream and downstream direction. We evaluate the fraction of time the voice quality is good in both directions. The traces are 1-hour long and from an area in *Amherst* with dense AP deployment.

Figure 6.21(a) shows VoIP performance as a function of switching delay threshold. We see that for values below 60 ms, Wiffler is as good as the *Oracle*. It does not hurt VoIP quality if we can discover within that time that WiFi will lose or delay the packet. Of the four systems, *No-switching* performs the worst because of high loss rates. Wiffler improves voice quality to 73%, compared to 41% when using the *No-*

switching system that switches to WiFi whenever available. Wiffler performs better than *Only-3G* because 3G frequently experiences high delays [75]. By dynamically deciding when to switch from WiFi to 3G, Wiffler provides both the low delays of WiFi and the high reliability of 3G.

Figure 6.21(b) shows that this advantage does come at the cost of a modest increase in 3G usage. Compared to *No-switching*, the increase in 3G usage is 10% if the switching delay threshold is 60 ms. Given the benefits of fast switching to application quality, we consider this increase to be a worthwhile trade-off.

6.6 Wiffler Conclusions

In this chapter, I establish the thesis statement (Section 1.1) in the context of cellular networks. Specifically, I show that leveraging opportunistic WiFi connectivity can reduce the spectrum pressure on 3G networks. Using a measurement study, we find that disruptions/unavailability in 3G networks can be significantly reduced using WiFi. But straightforward techniques of combining 3G and WiFi will not significantly reduce spectrum pressure on 3G, and may affect application performance. To overcome these challenges, we design Wiffler that substantially reduces 3G usage for delay-tolerant applications, while ensuring that the performance of loss-sensitive applications is not affected. Consistent with the thesis statement, we show that opportunistic protocols can be used to improve performance of cellular networks. However, unlike the protocols described in Chapters 3, 4, and 5, Wiffler is not a utility-driven protocol.

I answer the research questions raised in Section 1.2 with respect to cellular networks using measurement, protocol design, and testbed evaluation. We conduct a measurement study in 3 cities to understand the disruption and performance characteristics of 3G networks. We find that 3G is available 87% of the time, but combining WiFi and 3G can reduce disruptions/unavailability in 3G by 50%. However, our

study suggests that WiFi can at most offload 11% of 3G data. We also find that in half of the locations where WiFi is available, its throughput is much less than 3G. WiFi also experiences a much higher loss rate.

Our design goals in Wiffler is to address the availability and performance challenges of WiFi. Wiffler uses two key ideas to overcome these challenges: *leveraging delay tolerance* and *fast switching to 3G.* For delay tolerant applications, Wiffler trades off higher application latency for lower 3G spectrum usage. Instead of transmitting data immediately, it waits for WiFi to become available. Using a prediction algorithm to predict future WiFi throughput, it waits only if 3G savings are expected within the application's delay tolerance. For performance sensitive applications, Wiffler switches quickly to 3G when WiFi quality becomes poor, to maintain application quality.

We implement and deploy Wiffler on the Dome-3G testbed. We evaluate Wiffler using the deployment and using trace-driven simulations. In our deployment, we observed that for transfers of size 5MB that can be delayed by at most 60 seconds, Wiffler reduces 3G usage by 30%. In simulation using realistic workloads, we find that Wiffler reduces 3G usage by 45% for a 60 second delay tolerance. Because of it's wait-only-if-it-helps strategy, the actual transfer latency is increased by only 7 seconds on average. For a VoIP application, we find that the time periods with good VoIP quality increases by 31% (absolute improvement) using fast switching, compared to a system that switches to WiFi irrespective of its quality. More importantly, the increase in quality is achieved even when 40% of the VoIP traffic was sent over WiFi.

CHAPTER 7

FUTURE WORK

The vision for this thesis is to provide ubiquitous network access to mobile users in diverse environments. As a first step towards this vision, I designed robust protocols to overcome disruptions and enable applications in mobile networks. However, to make mobile access truly ubiquitous, two fundamental challenges remain, namely energy limitations and the lack of adaptability.

Energy limitation is the primary impediment to deploying resource-intensive applications on mobile devices. As mobile devices become more powerful, they have the potential to make pervasive computing a reality by providing support for applications including face recognition, speech-to-text translation, and health care applications. However, mobile devices cannot support several of these applications without quickly draining their battery.

The second challenge is the lack of adaptability. Users not only operate in diverse environments, but they also move between environments. For example, devices often go from being stationary to being mobile, or they move from mostly connected environments to intermittently-connected environments. However, mobile protocols and applications rarely adapt to these changes. For example, when a user moves to a poorly connected environment, protocols and applications simply stop working rather than degrading gracefully.

Below, I describe three research topics I wish to pursue as part of future work, to address the above challenges.

Energy benefits by exploiting radio diversity: The network interface is one of the main energy draws in mobile devices. The motivation for this work is that most mobile devices today are equipped with multiple interfaces corresponding to different radio technologies. Furthermore, the interfaces often have complimentary energy profiles, as I observed in my recent work [25, 44] and as corroborated by other researchers [91]. For example, transferring data on the WiFi interface is more energy efficient compared to 3G. On the other hand, keeping the WiFi interface powered on requires much more energy compared to 3G. The complimentary energy profiles suggest that combining the different interfaces will provide substantial energy benefits.

My goal is to exploit this diversity in radio technology to reduce energy consumed by the network interface. Specifically, I propose to design a virtual network interface that chooses the optimal physical interface to transmit data at a given instant. There are two challenges in exploiting diversity in radio technologies. First, it is non-trivial to choose the best interface at a given instant, since it may depend on several factors such as communication distance, loss rate, and the environment. Second, switching between interfaces can incur a performance penalty. In addition, several systems challenges will need to be addressed, including accommodating multiple IP end points and maintaining end-to-end semantics.

Adaptation policies to support seamless connectivity: Mobile devices operate in network environments that are constantly changing. However, protocol design decisions that are suitable for one environment are often ill-suited for a different environment. For example, opportunistic replication provides significant benefits for DTN routing [17] but is likely to increase congestion in mostly connected networks. My long term research agenda in this area is to design protocols that incorporate adaptability as a first-class design concern.

As a first step, my goal is to design rate adaptation policies as a device moves between stationary and mobile environments. Popular rate adaptation policies such

as SampleRate [82] vary the transmission rate according to the channel condition to adapt to losses in stationary environments. The challenge is that rate adaptation policies that work well in stationary environments do not work well in mobile environments because of the dynamic nature of the mobile channel. These challenges are further exacerbated in newer WiFi standards such as 802.11n [14] that have multiple antennae.

Characterizing user behavior: User behavior is key to the success of a mobile system. Today, it is difficult to design new protocols or evaluate the benefits of existing protocols with respect to real users because of a lack of understanding of mobile usage patterns. For example, the energy benefits of exploiting radio diversity (the project described above), depends on how often a user is in an environment with access to multiple network interfaces. One of my goals in this topic is to design a platform that will help evaluate the performance of protocols with respect to real users. As a first step, I will collect traces of real usage patterns and deploy a large scale mobile testbed to replay the traces. Several challenges will need to be addressed, including maintaining privacy, anonymizing traces, and monitoring the health of the collected traces.

CHAPTER 8

CONCLUSIONS

In this thesis, I present a suite of opportunistic protocols that overcome disruptions and enable applications in diverse network environments. Specifically, I focus on four network environments that span the connectivity spectrum, starting from mostly disconnected DTNs to mostly connected cellular networks. I show using detailed measurement study on two vehicular testbeds that each of these environments are prone to varying disruption characteristics that makes it difficult to support certain kinds of applications.

The primary challenge in overcoming disruptions in the diverse environment is *uncertainty*—uncertainty in topology, uncertainty in connectivity, and uncertainty in channel conditions. In this thesis, I show that exploiting resources opportunistically, i.e., using resources as they become available rather than planning for them *a priori*, allows protocols to work well under uncertainty. However, naively using opportunism can waste resources and hurt performance. Instead, I design utility-driven, probabilistic, protocols that address the resource management challenge to implement the following opportunistic mechanisms: *Replication, Aggressive prefetching, Opportunistic forwarding, and Opportunistic augmentation.*

I show how the four protocols enable applications in diverse network environments: 1) RAPID's replication routing enables bulk transfer applications in DTNs, 2) Thedu's aggressive prefetching enables web search in intermittently connected networks, 3) ViFi's opportunistic forwarding enables highly interactive applications such as VoIP in well connected mesh networks, and 4) Wiffler's opportunistic augmen-

tation reduces the demand on cellular network to improve application performance. Today, these applications cannot be supported in their respective environments, and even if supported, suffer from poor performance.

Finally, I present a detailed evaluation of the protocols using implementation and deployment experiments on two vehicular testbeds. The deployment experiments show that the protocols are practical and can be implemented in realistic usage environments. For a broader evaluation across a range of environmental factors, I also conduct evaluations using simulation experiments based on real-world traces. The deployment and simulation results show that the protocols significantly improve the performance of applications compared to the state-of-the-art, in their respective environments.

APPENDIX

RAPID

A.1 DTN hardness results

In Section 3.5, I presented two theorems to formalize hardness of DTN routing. Below, I provide a proof for the two theorems.

Any DTN routing algorithm has to deal with two uncertainties regarding the future: unpredictable meeting schedule and unpredictable workload. RAPID is a local algorithm that routes packets based on the marginal utility *heuristic* in the face of these uncertainties. In this section, we show two fundamental reasons that make the case for a heuristic approach to DTN routing. First, we prove that computing optimal solutions is hard even with complete knowledge about the environment. Second, we prove that the presence of even one of the two uncertainties rule out provably efficient online routing algorithms.

A.1.1 Competitive Hardness of Online DTN Routing

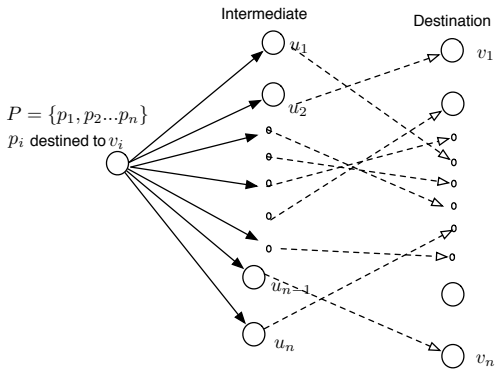


Figure A.1. DTN node meetings for Theorem A. Solid arrows represent node meetings known a priori to the online algorithm while dotted arrows represent meetings revealed subsequently by an offline adversary.

Let ALG be any deterministic online DTN routing algorithm with unlimited computational power.

THEOREM 1(a). If ALG has complete knowledge of the workload, but not of the schedule of node meetings, then ALG is $\Omega(n)$-competitive with an offline adversary with respect to the fraction of packets delivered, where n is the number of packets in the workload.

Proof. We prove the theorem by constructing an offline adversary, ADV, that incrementally generates a node meeting schedule after observing the actions of ALG at each step. We show how ADV can construct a node meeting schedule such that ADV can deliver all packets while ALG, without prior knowledge of node meetings, can deliver at most 1 packet.

Consider a DTN as illustrated in Fig. A.1, where $P = \{p_1, p_2, \ldots, p_n\}$ denotes a set of unit-sized packets; $U = \{u_1, u_2, \ldots, u_n\}$ denotes a set of intermediate nodes; and $V = \{v_1, v_2, \ldots, v_n\}$ denotes a set of nodes to which the packets are respectively destined, i.e. p_i is destined to v_i for all $i \in [1, n]$. The following procedure describes ADV's actions given ALG as input.

PROCEDURE FOR ADV:

- **Step 1**: ADV generates a set of node meetings involving unit-size transfer opportunities at time $t = 0$ between A and each of the intermediate nodes $u_1, \ldots, u_n$ respectively (refer to Figure A.1).

- **Step 2**: At time $t_1 > 0$, ADV observes the set of transfers X made by ALG. Without loss of generality, $X : P \rightarrow U$ is represented as a (one-to-many) mapping where $X(p_i)$ is the set of intermediate nodes $(u_1, u_2 \cdots u_n)$ to which ALG replicates packet p_i.

- **Step 3**: ADV generates the next set of node meetings $(u_1, Y(u_1)), (u_2, Y(u_2)), \ldots, (u_n, Y(u_n))$ at time t_1, where $Y : U \rightarrow V$ is a bijective mapping from the set of intermediate nodes to the destination nodes $v_1, v_2, \cdots v_n$.

ADV uses the following procedure to generate the mapping Y given X in Step 3.

PROCEDURE GENERATE_Y(X):

```
1 Initialize Y(p_i) to null for all i ∈ [1, n];
2 for each i ∈ [1, n] do
3   if ∃j : u_j ∉ X(p_i) and Y(u_j) = null, then
4     Map Y(u_j) → v_i for the smallest such j;
5   else
6     Pick a j: Y(u_j) = null, and map Y(u_j) → v_i
7   endif
```

LEMMA 1. *ADV executes Line 6 in* GENERATE_Y(X) *at most once.*

Proof. We first note that the procedure is well defined at Line 6: each iteration of the main loop map exactly one node in U to a node in V, therefore a suitable j such that $Y(u_j) = \text{null}$ exists. Suppose ADV first executes Line 6 in the m'th iteration. By inspection of the code, the condition in Line 3 is false, therefore each intermediate node u_k, $k \in [1, n]$, either belongs to $X(p_i)$ or is mapped to some destination node $Y(u_k) \neq \text{null}$. Since each of the $m-1$ previous iterations must have executed Line 4 by assumption, exactly $m-1$ nodes in U have been mapped to nodes in V. Therefore, each of the remaining $n - m + 1$ unmapped nodes must belong to $X(p_i)$ in order to falsify Line 3. Line 6 maps one of these to v_i leaving $n - m$ unmapped nodes. None of these $n - m$ nodes is contained in $X(p_k)$ for $k \in [m + 1, \ldots, n]$. Thus, in each of the subsequent $n - m$ iterations, the condition in Line 3 evaluates to true. □

LEMMA 2. *The schedule of node meetings created by Y allows ALG to deliver at most one packet to its destination.*

Proof. For ALG to deliver any packet p_i successfully to its destination v_i, it must be the case that some node in $X(p_i)$ maps to v_i. Such a mapping could not have occurred in Line 3 by inspection of the code, so it must have occurred in Line 6. By Lemma 1, Line 6 is executed exactly once, so ALG can deliver at most one packet. □

LEMMA 3. *The schedule of node meetings created by Y allows ADV to deliver all packets to their respective destinations.*

Proof. We first note that, by inspection of the code, Y is a bijective mapping: Line 4 and 6 map an unmapped node in U to v_i in iteration m and there are n such iterations. So, ADV can route p_i by sending it $Y^{-1}(v_i)$ and subsequently to v_i. □

Theorem 1(a) follows directly from Lemmas 2 and 3.

□

COROLLARY 1. *ALG can be arbitrarily far from ADV with respect to average delivery delay.*

Proof. The average delivery delay is unbounded for ALG because of undelivered packets in the construction above while it is finite for ADV. If we assume that that ALG can eventually deliver all packets after a long time T (say, because all nodes connect to a well-connected wired network at the end of the day), then ALG is $\Omega(T)$-competitive with respect to average delivery delay using the same construction as above. □

We remark that it is unnecessary in the construction above for the two sets of n node meetings to occur simultaneously at $t = 0$ and $t = t_1$, respectively. The construction can be easily modified to not involve any concurrent node meetings.

THEOREM 1(b). If ALG has complete knowledge of the meeting schedule, but not of

the packet workload, then ALG can deliver at most a third of the packets delivered by an optimal offline adversary.

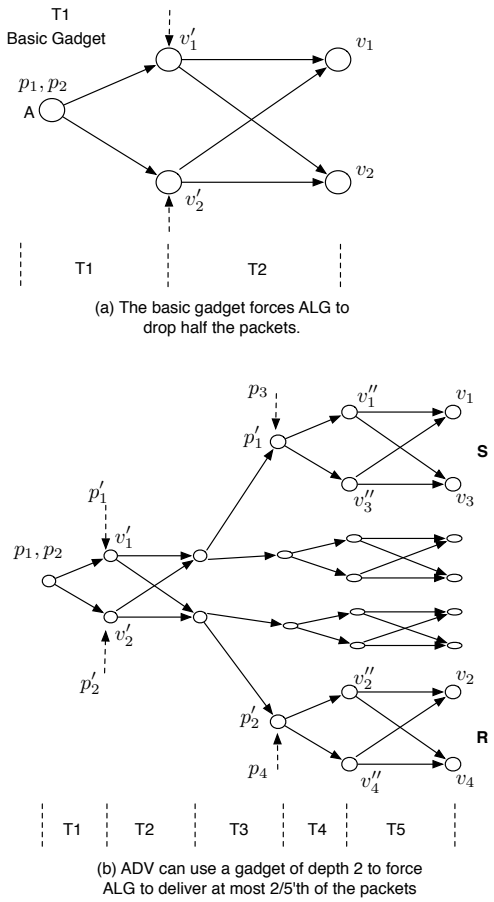


Figure A.2. DTN construction for Theorem A. Solid arrows represent node meetings known a priori to ALG while vertical dotted arrows represent packets created by ADV at the corresponding node.

Proof. We prove the theorem by constructing a procedure for ADV to incrementally generate a packet workload by observing ALG's transfers at each step. As before, we only need unit-sized transfer opportunities and packets for the construction.

Consider the basic DTN "gadget" shown in Fig. A.2(a) involving just six node meetings. The node meetings are known in advance and occur at times T_1 and $T_2 > T_1$ respectively. The workload consists of just two packets $P = \{p_1, p_2\}$ destined to v_1 and v_2, respectively.

LEMMA 4. *ADV can use the basic gadget to force ALG to drop half the packets while itself delivering all packets.*

Proof. The procedure for ADV is as follows. If ALG transfers p_1 to v_1' and p_2 to v_2', then ADV generates two more packets: p_2' at v_1' destined to v_2 and p_1' at v_2' destined to v_1. ALG is forced to drop one of the two packets at both v_1' and v_2'. ADV can deliver all four packets by transferring p_1 and p_2 to v_2' and v_1' respectively at time T_1, which is the exact opposite of ALG's choice.

If ALG instead chooses to transfer p_1 to v_2' and p_2 to v_1', ADV chooses the opposite strategy.

If ALG chooses to replicate one of the two packets in both transfer opportunities at time T_1 while dropping the other packet, ADV simply deliver both packets. Hence the lemma. □

Next, we extend the basic gadget to show that ALG can deliver at most a third of the packets while ADV delivers all packets. The corresponding construction is shown in Figure A.2(b).

The construction used by ADV composes the basic gadget repeatedly for a depth of 2. In this construction, ADV can force ALG to drop 2/5th of the packet while ADV delivers all packets. We provide the formal argument in a technical report [18] in the interest of space. Similarly, by creating a gadget of depth 3, we can show that ADV can force ALG to deliver at most 4/11'th of the packets. Effectively, each new basic gadget introduces 3 more packets and forces ALG to drop 2 more packets. In particular, with a gadget of depth i, ADV can limit ALG's delivery rate to $i/(3i-1)$. Thus, by composing a sufficiently large number of basic gadgets, ADV can limit the delivery rate of ALG to a value close to 1/3. □

A.1.2 Computational Hardness of the DTN Routing Problem

THEOREM 1 (A): Given complete knowledge of node meetings and the packet workload *a priori*, computing a routing schedule that is optimal with respect to the number of packets delivered is NP-hard and has a lower bound of $\Omega(n^{1/2-\epsilon})$ on the approximation ratio.

Proof. Consider a DTN routing problem with n nodes that have complete knowledge of node meetings and workload *a priori*. The input to the DTN problem is the set of nodes $1, \ldots, n$; a series of transfer opportunities $\{(u_1, v_1, s_1, t_1), (u_2, v_2, s_2, t_2), \ldots\}$ such that $u_i, v_i \in [1, n]$, s_i is the size of the transfer opportunity, and t_i is the time of meeting; and a packet workload $\{p_1, p_2, \ldots p_s\}$, where $p_i = (u_i', v_i', s_i', t_i')$, where $u', v' \in [1, n]$ are the source and destination, s' the size, and t' the time of creation of the packet, respectively. The goal of a DTN routing algorithm is to compute a feasible schedule of packet transfers, where *feasible* means that the total size of transferred packets in any transfer opportunity is less than the size of the transfer opportunity.

The decision version $O_{n,k}$ of this problem is: Given a DTN with n nodes such that nodes have complete knowledge of transfer opportunities and the packet workload, is there a feasible schedule that delivers at least k packets?

LEMMA 5. *$O(n, k)$ is NP-hard.*

Proof. We show that $O(n, k)$ is a NP-hard problem using a polynomial-time reduction from the edge-disjoint path (EDP) problem for a directed acyclic graph (DAG) to $O(n, k)$. The EDP problem for a DAG is known to be NP-hard [39].

The decision version of EDP problem is: Given a DAG $G = (V, E)$, where $|V| = n$, $E \in V \times V$: $e_i = (u_i, v_i) \in E$, if e_i is incident on u_i and v_i and direction is from u_i to v_i. If given source-destination pairs $\{(s_1, t_1), (s_2, t_2)...(s_s, t_s)\}$, do a set of edge-disjoint paths $\{c_1, c_2...c_k\}$ exist, such that c_i is a path between s_i and t_i, where $1 \leq i \leq k$.

Given an instance of the EDP problem, we generate a DTN problem $O(n, k)$ as follows:

As the first step, we topologically order the edges in G, which is possible given G is a DAG. The topological sorting can be performed in polynomial-time.

Next, we label edges using natural numbers with any function $l : E \rightarrow N$ such that if $e_i = (u_i, u_j)$ and $e_j = (u_j, u_k)$, then $l(e_i) < l(e_j)$. There are many ways to define such a function l. One algorithm is:

1 label = 0

2 For each vertex v in the decreasing order of the topological sort,

(a) Choose unlabeled edge $e = (v, x) : x \in V$,

(b) label = label + 1

(c) Label e; l(e) = label.

Since vertices are topologically sorted, if $e_i = (u_i, u_j)$ then $u_i < u_j$. Since the algorithm labels all edges with source u_i before it labels edges with source u_j, if $e_j = (u_j, u_k)$, then $l(e_i) < l(e_j)$.

Given a G, we define a DTN routing problem by mapping V to the nodes $(1, .., n)$ in the DTN. The edge $(e = \{u, v\} : u, v \in V)$ is mapped to the transfer opportunity $(u, v, 1, l(e))$, assuming transfer opportunities are unit-sized. Source and destination pairs $\{(s_1, t_1), (s_2, t_2), \ldots, (s_m, t_m)\}$ are mapped to packets $\{p_1, p_2, \ldots, p_m\}$, where $p_i = (s_i, t_i, 1, 0)$. In other words, packet p is created between the corresponding source-destination pair at time 0 and with unit size. A path in graph G is a valid route in the DTN because the edges on a path are transformed to transfer opportunities of increasing time steps. Moreover, a transfer opportunity can be used to send no more than one packet because all opportunities are unit-sized. If we solve the DTN routing problem of delivering k packets, then there exists k edge-disjoint paths in graph G, or in other words we can solve the EDP problem. Similarly, if the EDP problem has a solution consisting of k edge-disjoint paths in G, at least k packets can be delivered

using the set of transfer opportunities represented by each path. Using the above polynomial-time reduction, we show that a solution to EDP exists if and only if a solution to $O(n, k)$ exists. Thus, $O(n, k)$ is NP-hard. □

COROLLARY 2. *The DTN routing problem has a lower bound of* $\Omega(n^{1/2-\epsilon})$ *on the approximation ratio.*

Proof. The reduction given above is a true reduction in the following sense: each successfully delivered DTN packet corresponds to an edge-disjoint path and vice-versa. Thus, the optimal solution for one exactly corresponds to an optimal solution for the other. Therefore, this reduction is an L-reduction [89]. Consequently, the lower bound $\Omega(n^{1/2-\epsilon})$ known for the hardness of approximating the EDP problem [57] holds for the DTN routing problem as well. □

Hence, Theorem 2. □

The hardness results naturally extend to the average delay metric for both the online as well as computationally limited algorithms.

A. 2 Delay estimation based on dependency graphs

In Section 3, we presented ESTIMATE_DELAY that estimates expected delays of packets based on the packet's position in a node's buffer. The algorithm ignores some dependencies between packets across node buffers. In this section, we present an algorithm DAG_DELAY to estimate expected delays more accurately without ignoring non vertical dependancies.

To formalize the dependancies, we introduce some notation. Let $G = (V, E)$ be a graph representing a markov network with vertices $V = \{V_1 \cup V_2 \cup \ldots \cup V_m\}$ where $V_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,k}\}$ is the set of k replicas of packet i. All packets in V are destined to the same DTN node — recall that we wish to estimate expected delays of packets based on the current state of the network assuming no further replication, so packets destined to other DTN nodes do not affect the delays of packets in V.

An edge (or a path) from one vertex to another indicates a dependency between the delivery time distributions of the corresponding packets. The edges are constructed as follows.

- Each replica is connected to its successor, i.e., the replica immediately ahead of it in the current buffer.
- Each replica is connected to all the replicas of its successor at other DTN node buffers.

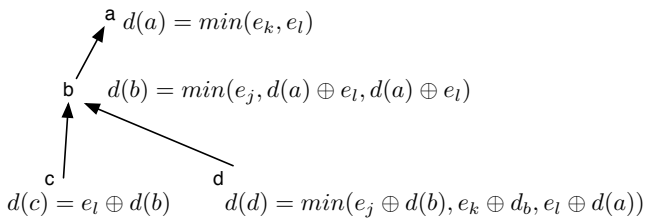


Figure A.3. A topologically sorted dependancy graph.

Let the delay distribution of a packet in buffer x be e_x. Let $\oplus$ represent the addition of two distributions (e.g., adding two identical exponential distributions yields a gamma distribution with twice the mean). Assume unit-sized transfer opportunity and packet.

DAG_DELAY first topologically sorts the dependancy graph. For example, the topological sort of Figure 3.4 is shown in Figure A.3. DAG_DELAY computes the delay of the packets in the graph in the topologically order starting from the top. The information maintained for each of the k replicas $p_1, \ldots, p_k$ of packet p is $\{succ(p_j), e_{vertex(p_j)}\}$, $1 \leq j \leq k$, where $succ(p_j)$ is the successor of the replica p_j, and $vertex(p_j)$ is the DTN buffer where p_j exists.

PROCEDURE DAG_DELAY(p):

1 for each replica $p_j, 1 \leq j \leq k$ of p, do

(a) Let $s = succ(p_j)$, and $n = vertex(p_j)$

(b) if $d(s)$ is not defined, then

i. $d(s) =$DAG_DELAY(s)

(c) $d'(p_j) = d(s) \oplus e_n$

2 return $d(p) = min(d'(p_1), \ldots, d'(p_k))$

Figure A.3 presents the delay of each packet as computed using DAG_DELAY. Although the algorithm is recursive, sequentially computing the delay of packets top down in the DAG and storing the delay values of already computed packets ensures that the delay of each packet is computed exactly once. And since the DAG has no cycles, DAG_DELAY will converge.

DAG_DELAY is an idealized algorithm and its implementation requires a global control channel with complete knowledge of the system state. Therefore, in our implementation, we use the less accurate but local ESTIMATE_DELAY algorithm. In addition, DAG_DELAY fails when the transfer opportunities are not unit-sized. For example, in Figure 3.4, if the transfer opportunity and packets are unit-sized, then the delay of packet b depends on the delay of packet a. But if not, then the delay of b may not depend on the delay of a and the dependancy graph is no longer valid. In general, estimating the expected delay of packets is a hard problem even when global knowledge is available but transfer opportunities are *not* unit-sized.

Pathological examples when ESTIMATE_DELAY *fails*

ESTIMATE_DELAY ignores non-vertical edges in the dependency DAG and therefore is not an accurate estimate. Figure A.4 shows a pathological case of packet distribution where the estimation error of ESTIMATE_DELAY can be arbitrarily large. In this example, we assume that all packets are destined to node Z. There are $k+1$ replicas of a distributed among nodes $W_1, W_2 \cdots W_k$ and X. There are no replicas of b. We assume that the meeting times between all pairs of nodes is exponentially

distributed; further we let the mean meeting time between $W_1, W_2 \cdots W_k$ and Z be λ and between X and Z be $10 \cdot \lambda$. The transfer opportunities are unit-sized.

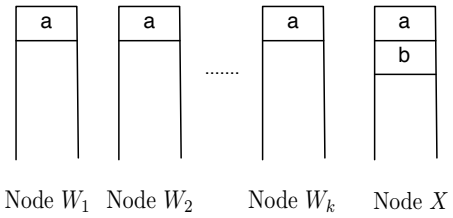


Figure A.4. A pathological example of packet distribution among nodes.

Algorithm ESTIMATE_DELAY computes the delay of delivery b as the expected time taken for X to meet destination Z twice. Accordingly, a RAPID node estimates the delivery delay of b as an exponential distribution with mean $20 \cdot \lambda$.

DAG_DELAY, on the other hand, takes into account non-vertical dependancies and estimates the delivery delay distribution of b accurately as – the time taken for one of the $k+1$ replicas of a to be delivered followed by X meeting Z. The time taken for a to be delivered is an exponential distribution with mean $\frac{\lambda}{k}$. Accordingly, DAG_DELAY estimates the delivery time of b as a gamma distribution with mean $10 \cdot \lambda + \frac{\lambda}{k}$.

The difference is delay estimation between DAG_DELAY and ESTIMATE_DELAY can be arbitrarily large in this pathological example. But notwithstanding such pathological scenarios, ESTIMATE_DELAY is simple, local, and computationally efficient heuristic to estimate expected delays and we find that it works well in practice.

A.3 ILP formulation of DTN routing problem

In Section 3.6.2.4 we compare the performance of RAPID with an optimal routing protocol that solves the DTN routing problem with complete future knowledge. We formulate the offline, optimal, DTN routing problem as an Integer Linear Program (ILP) optimization problem when the meeting times between nodes are precisely know. We divide time into discrete intervals so every node meets at most one other node in an interval. Jain et al. [62] solve a similar DTN routing problem but allow

packets to be fragmented across links and mapped non-zero propagation delays on the links. This severely limited the size of the network and the number of packets they could evaluate. In comparison, our formulation lets us obtain the optimal solution for realistic DTNs with small to moderate workloads.

The inputs to the problem are as follows.

- The set of time intervals $I = 1, 2, \ldots, h$. The function b returns the beginning of the interval. e returns the end of an interval and variable h represents the last interval
- The set of nodes in the network N
- The set of edges E. An edge is defined when two nodes meeting in an interval. We define functions f and s to return the first and the second node that meet respectively, d returns the interval in which the edge is defined. When two nodes i and j meet, they are represented two edges e and e' on either direction. $E_{(x,y)}$ represents an edge with source x and destination y.
- The set of packets P. Function st return the source of the packet, dt return the destination of the packet, c returns the interval in which the packet was created, t returns time the packet was created and $size()$ returns the size of the packet.
- The bandwidth for each meeting is a constant and is B.

The variables are

- $X(p \in P, e \in E) = 1$ if j is forwarded over the edge e and is 0 otherwise
- $N(p \in P, n \in N, i \in I) = 1$ if node n has packet p in the interval i and is 0 otherwise
- $D(p \in P, i \in I) = 1$ if packet p is delivered before interval i and is 0 otherwise

X can be used to construct the optimal path taken by a packet.

$$min \sum_{p \in P} \sum_{i \in I} \sum_{e \in E_{(dt(p),i)}} (b(i) - t(p)) \cdot X(p,e) + \sum_{p \in P} (1 - D(p, e(h)) \cdot (b(h) - t(p))$$

All constraints use notations $\forall p, n, i, e$ to mean $\forall p \in P, \forall n \in N, \forall i \in I$ and $\forall e \in E$. The constraints are

Initialization constraints

$N(p,n,i) = 0 \ if \ i < c(p) \ \forall p, n, i$

$N(p,n,i) = 1 \ if \ st(p) = n$ and $c(p) = i \ \forall p$

Bandwidth constraint

$$\sum_{p \in P} (X(p,e) * size(p) \leq B \ \forall e$$

Transfer constraints

$$N(p,n,i-1) - \sum_{e \in E_{(i,n)}} X(p,e) \ \forall \ p,n,i$$

$$\sum_{e \in E_{(n,i)}} X(p,e) - N(p,n,i) = 0 \ \forall p,n,i$$

$N(p, f(e), d(e) - 1) - X(p,e) >= 0 \ \forall p, e$

Conservation constraint

$$1 - \sum_{n \in N} N(p,n,i) = 0 \ if \ i > c(p) \ \forall p, e$$

Delivery Constraint

$$D(p,i) - \sum_{e \in E_{(dt(p),.)} : d(e) < i} X(p,e) = 0 \ \forall \ p, i$$

BIBLIOGRAPHY

[1] Google wifi. http://en.wikipedia.org/wiki/Google_WiFi.

[2] One laptop per child. http://www.laptop.org.

[3] Optimized link state routing protocol. http: //www.olsr.org/.

[4] Qualnet. http://www.scalable-networks.com/products.

[5] Text Retrieval Conference (TREC). http://trec.nist.gov.

[6] TIER Project, UC Berkeley. http://tier.cs.berkeley.edu/.

[7] Umass dome testbed. http://prisms.cs.umass.edu/dome.

[8] Umass dome testbed. http://prisms.cs.umass.edu/dome.

[9] Umass turtlenet. http://prisms.cs.umass.edu/dome/turtlenet.

[10] White spaces. http://en.wikipedia.org/wiki/White_spaces_(radio)/.

[11] Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. 1936–1948.

[12] EDGE, HSPA and LTE Broadband Innovation. http://www.3gamericas.org/documents/EDGE_HSPA_and_LTE_Broadband_Innovation_Rysavy_Sept_2008.pdf/, 2010.

[13] Mobile Broadband Capacity Constraints And the Need for Optimization. http://www.rysavy.com/Articles/2010_02_Rysavy_Mobile_Broadband_Capacity_Constraints.pdf, 2010.

[14] 802.11n report. http://grouper.ieee.org/groups/802/11/Reports/tgn_update.htm.

[15] Aguayo, Daniel, Bicket, John, Biswas, Sanjit, Judd, Glenn, and Morris, Robert. Link-level measurements from an 802.11b mesh network. In *SIGCOMM* (Aug. 2004).

[16] Amir, Yair, Danilov, Claudiu, Hilsdale, Michael, Musaloiu-Elefteri, Raluca, and Rivera, Nilo. Fast handoff for seamless wireless mesh networks. In *MobiSys* (June 2006).

[17] Balasubramanian, Aruna, Levine, Brian Neil, and Venkataramani, Arun. DTN Routing as a Resource Allocation Problem. In *Proc. ACM SIGCOMM* (August 2007), pp. 373–384.

[18] Balasubramanian, Aruna, Levine, Brian Neil, and Venkataramani, Arun. DTN Routing as a Resource Allocation Problem. Tech. Rep. 07-37, UMass Amherst, 2007.

[19] Balasubramanian, Aruna, Levine, Brian Neil, and Venkataramani, Arun. Enabling Interactive Applications in Hybrid Networks. In *Proc. ACM Mobicom* (September 2008).

[20] Balasubramanian, Aruna, Levine, Brian Neil, and Venkataramani, Arun. Replication Routing in DTNs: A Resource Allocation Approach. *IEEE/ACM Transactions on Networking 18*, 2 (April 2010), 596–609.

[21] Balasubramanian, Aruna, Mahajan, Ratul, and Venkataramani, Arun. Augmenting mobile 3g using wifi. In *MobiSys '10: Proceedings of the 8th international conference on Mobile systems, applications, and services* (New York, NY, USA, 2010), ACM, pp. 209–222.

[22] Balasubramanian, Aruna, Mahajan, Ratul, Venkataramani, Arun, Levine, Brian Neil, and Zahorjan, John. Interactive WiFi Connectivity for Moving Vehicles. In *Proc. ACM SIGCOMM* (August 2008).

[23] Balasubramanian, Aruna, Mahajan, Ratul, Venkataramani, Arun, Levine, Brian Neil, and Zahorjan, John. Interactive wifi connectivity for moving vehicles. Tech. Rep. TR-2008-18, Dept. of Computer Science, University of Massachusetts, 2008.

[24] Balasubramanian, Aruna, Zhou, Yun, Croft, W. Bruce, Levine, Brian Neil, and Venkataramani, Arun. Web Search From a Bus. In *Proc. ACM Workshop on Challenged Networks (CHANTS)* (September 2007), pp. 59–66.

[25] Balasubramanian, Niranjan, Balasubramanian, Aruna, and Venkataramani, Arun. Energy consumption in mobile phones: a measurement study and implications for network applications. In *IMC '09: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference* (New York, NY, USA, 2009), ACM, pp. 280–293.

[26] Banerjee, Nilanjan, Corner, Mark D., and Levine, Brian Neil. An Energy-Efficient Architecture for DTN Throwboxes. In *Proceedings of IEEE Infocom* (Anchorage, Alaska, May 2007), pp. 776–784.

[27] Biswas, Sanjit, and Morris, Robert. ExOR: opportunistic multi-hop routing for wireless networks. In *SIGCOMM* (Aug. 2005).

[28] Brik, Vladimir, Mishra, Arunesh, and Banerjee, Suman. Eliminating handoff latencies in 802.11 WLANs using multiple radios: Applications, experience, and evaluation. In *IMC* (Oct. 2005).

[29] Broder, Andrei. A taxonomy of web search. *SIGIR Forum 36*, 2 (2002), 3–10.

[30] Buddhikot, M., Chandranmenon, G., S.J.Han, Y.W.Lee, and amd L.Salgarelli, S.Miller. Integration of 802.11 and Third Generation Wireless Data Networks. In *Proc. IEEE Infocom* (April 2003).

[31] Burgess, John, Gallagher, Brian, Jensen, David, and Levine, Brian Neil. MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks. In *Proc. IEEE INFOCOM* (April 2006).

[32] Burns, Brendan, Brock, Oliver, and Levine, Brian Neil. Autonomous Enhancement of Disruption Tolerant Networks. In *Proc. IEEE International Conference on Robotics and Automation* (May 2006).

[33] Camp, Joseph D., Knightly, Edward W., and Reed, William S. Developing and deploying multihop wireless networks for low-income communities. In *in Proceedings of Digital Communities* (2005).

[34] Cao, Y., and Li., V. Scheduling algorithms in broadband wireless networks. vol. 1, pp. 76–87.

[35] Casella, George, and Berger, Roger L. *Statistical Inference*. Second Edition. Duxbury, 2002.

[36] Chachulski, Szymon, Jennings, Michael, Katti, Sachin, and Katabi, Dina. Trading structure for randomness in wireless opportunistic routing. In *SIGCOMM* (Aug. 2007).

[37] Chaintreau, A., Hui, P., Crowcroft, J., Diot, C., Gass, R., and Scott, J. Impact of Human Mobility on the Design of Opportunistic Forwarding Algorithms. In *Proc. IEEE Infocom* (May 2006).

[38] Chandra, Bharat, Dahlin, Mike, Gao, Lei, Khoja, Amjad-Ali, Nayate, Amol, Razzaq, Asim, and Sewani, Anil. Resource Management for Scalable Disconnected Access to Web Services. In *Proc. Intl World Wide Web Conf.* (May 2001), pp. 245–256.

[39] Chekuri, Chandra, Khanna, Sanjeev, and Shepherd, F. Bruce. An $O(\sqrt{(}n))$ Approximation and Integrality Gap for Disjoint Paths and Unsplittable Flow. *Theory of Computing 2*, 7 (2006), 137–146.

[40] Chen, Binbin, and Chan, Mun Choon. MobTorrent: A Framework for Mobile Internet Access from Vehicles. In *Proc. ACM Infocom* (April 2009).

[41] Cole, R. G., and Rosenbluth, J. H. Voice over IP performance monitoring. *CCR 31*, 2 (2001).

[42] Couto, D. D., Aguayo, D., Bicket, J., and Morris, R. A high-throughput path metric for multi-hop wireless routing. In *MobiCom* (Sept. 2003).

[43] CPLEX. `http://www.ilog.com`.

[44] Cuervo, Eduardo, Balasubramanian, Aruna, Cho, Dae-ki, Wolman, Alec, Saroiu, Stefan, Chandra, Ranveer, and Bahl, Paramvir. Maui: making smartphones last longer with code offload. In *MobiSys '10: Proceedings of the 8th international conference on Mobile systems, applications, and services* (New York, NY, USA, 2010), ACM, pp. 49–62.

[45] Demmer, Michael, and Fall, Kevin. Dtlsr: delay tolerant routing for developing regions. In *NSDR '07: Proceedings of the 2007 workshop on Networked systems for developing regions* (New York, NY, USA, 2007), ACM, pp. 1–6.

[46] Deshpande, Pralhad, Kashyap, Anand, Sung, Chul, and Das, Samir R. Predictive methods for improved vehicular wifi access. In *Proc. MobiSys '09* (June 2009).

[47] Ietf delay tolerant network research group.

[48] Eriksson, Jakob, Girod, Lewis, Hull, Bret, Newton, Ryan, Madden, Samuel, and Balakrishnan, Hari. The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring. In *The Sixth Annual International conference on Mobile Systems, Applications and Services (MobiSys 2008)* (Breckenridge, U.S.A., June 2008).

[49] Eriksson, Jakob, Madden, Sam, and Balakrishnan, Hari. Cabernet: A Content-Delivery Network for Moving Vehicles. In *Proc. ACM Mobicom* (September 2008).

[50] Gallager, R. A Minimum Delay Routing Algorithm Using Distributed Computation. In *IEEE Trans. on Communications* (Jan 1977), vol. 25, pp. 73–85.

[51] Garg, Nitin, Sobti, Sumeet, Lai, Junwen, Zheng, Fengzhou, Li, Kai, Krishnamurthy, Arvind, and Wang, Randolph. Bridging the Digital Divide. *ACM Trans. on Storage 1*, 2 (May 2005), 246–275.

[52] Gass, Richard, Scott, James, and Diot, Christophe. Measurements of in-motion 802.11 networking. In *Workshop on Mobile Computing Systems and Applications (WMSCA)* (Apr. 2006).

[53] Giannoulis, Anastasios, Fiore, Marco, and Knightly, Edward W. Supporting vehicular mobility in urban multi-hop wireless networks. In *MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services* (New York, NY, USA, 2008), ACM, pp. 54–66.

[54] Giannoulis, Anastasios, Fiore, Marco, and Knightly, Edward W. Supporting Vehicular Mobility in Urban Multi-hop Wireless Networks. In *Proc. MobiSys* (June 2008).

[55] Giordano, Silvia, Lenzarini, Davide, Puiatti, Alessandro, and Vanini, Salvatore. Enhanced DHCP client. Demo at CHANTS, Sept. 2007.

[56] Goodman, D. J., Borras, J., Mandayam, N. B., and Yates, R. D. Infostations: A New System for Data and Messaging Siervices. In *Proc. Vehicular Technology Conference* (May 1997), pp. 969–973.

[57] Guruswami, Venkatesan, Khanna, Sanjeev, Rajaraman, Rajmohan, Shepherd, Bruce, and Yannakakis, Mihalis. Near-Optimal Hardness Results and Approximation Algorithms for Edge-Disjoint Paths and Related Problems. In *Proc. ACM STOC* (1999), pp. 19–28.

[58] Hadaller, David, Keshav, Srinivasan, Brecht, Tim, and Agarwal, Shubham. Vehicular opportunistic communication under the microscope. In *MobiSys* (June 2007).

[59] Hadaller, David, Keshav, Srinivasan, Brecht, Tim, and Agarwal, Shubham. Vehicular Opportunistic Communication Under the Microscope. In *Proc. ACM Mobisys* (June 2007), pp. 206–219.

[60] ho Shin, Min, Mishra, Arunesh, and Arbaugh, William. Improving the latency of 802.11 hand-offs using neighbor graphs. In *MobiSys* (June 2004).

[61] Hull, Bret, et al. CarTel: A Distributed Mobile Sensor Computing System. In *Proc. ACM SenSys* (Oct. 2006), pp. 125–138.

[62] Jain, Sushant, Demmer, Michael, Patra, Rabin, and Fall, Kevin. Using Redundancy to Cope with Failures in a Delay Tolerant Network. In *Proc. ACM Sigcomm* (August 2005), pp. 109–120.

[63] Jain, Sushant, Fall, Kevin, and Patra, Rabin. Routing in a Delay Tolerant Network. In *Proc. ACM Sigcomm* (Aug. 2004), pp. 145–158.

[64] Jansen, B.J., Spink, A., and Saracevic, T. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing and Management 36*, 2 (2000), 207–227.

[65] Jiang, Z., and Kleinrock, L. Web prefetching in a mobile environment. In *IEEE Personal Communications* (September, 1998), vol. 5, pp. 25–34.

[66] Johnson, David B, and Maltz, David A. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, Imielinski and Korth, Eds., vol. 353. Kluwer Academic Publishers, 1996.

[67] Jones, Evan, Li, Lily, and Ward, Paul. Practical Routing in Delay-Tolerant Networks. In *Proc. ACM Chants Workshop* (Aug. 2005), pp. 237–243.

[68] Kelly, F., Maulloo, A., and Tan, D. Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability. In *J. Op. Res. Society* (1998), vol. 49, pp. 237–252.

[69] Kotz, David, Newport, Calvin, and Elliott, Chip. The mistaken axioms of wireless-network research. Tech. rep., Dartmouth College, July 2003.

[70] Leguay, Jeremie, Friedman, Timur, and Conan, Vania. DTN Routing in a Mobility Pattern Space. In *Proc. ACM Chants Workshop* (Aug. 2005), pp. 276–283.

[71] Leguay, Jeremie, Lindgren, Anders, Scott, James, Friedman, Timur, and Crowcroft, Jon. Opportunistic content distribution in an urban setting. In *CHANTS '06: Proceedings of the 2006 SIGCOMM workshop on Challenged networks* (New York, NY, USA, 2006), ACM, pp. 205–212.

[72] Leung, V., and Au, A. A wireless local area network employing distributed radio bridges. *ACM/Baltzer Wireless Network Journal 2* (1995).

[73] Li, Ming, Yan, Tingxin, Ganesan, Deepak, Lyons, Eric, Shenoy, Prashant, Venkataramani, Arun, and Zink, Michael. Multi-user Data Sharing in Radar Sensor Networks. In *Proc. ACM SenSys* (November 2007), pp. 247–260.

[74] Lindgren, Anders, Doria, Avri, and Schelén, Olov. Probabilistic Routing in Intermittently Connected Networks. In *Proc. SAPIR Workshop* (Aug. 2004), pp. 239–254.

[75] Liu, Xin, Sridharan, Ashwin, Machiraju, Sridhar, Seshadri, Mukund, and Zang, Hui. Experiences in a 3G Network: Interplay between the Wireless Channel and Applications. In *Proc. MobiCom* (September 2008).

[76] Mahajan, Ratul, Zill, Brian, and Zahorjan, John. Understanding WiFi-based connectivity from moving vehicles. In *IMC* (Nov. 2007).

[77] Manmatha, R., Rath, T., and Feng, F. Modeling score distributions for combining the outputs of search engines. In *Proc. ACM SIGIR* (2001), pp. 267–275.

[78] Mitchener, W.G., and Vadhat, A. Epidemic Routing for Partially Connected Ad hoc Networks. Tech. Rep. CS-2000-06, Duke Univ., 2000.

[79] Miu, Allen, Tan, Godfrey, Balakrishnan, Hari, and Apostolopoulos, John. Divert: fine-grained path selection for wireless LANs. In *MobiSys* (June 2004).

[80] Miu, Allen K., Balakrishnan, Hari, and Koksal, Can E. Improving Loss Resilience with Multi-Radio Diversity in Wireless Networks. In *MobiCom* (Sept. 2005).

[81] Moon, S. B., Skelly, P., and Towsley, D. Estimation and removal of clock skew from network delay measurements. In *INFOCOM* (Mar. 1999).

[82] Morris, Robert T., and Bicket, John C. Bit-rate selection in wireless networks. Tech. rep., Masters thesis, MIT, 2005.

[83] Navda, Vishnu, Subramanian, Anand Prabhu, Dhanasekaran, Kannan, Timm-Giel, Andreas, and Das, Samir. MobiSteer: Using directional antenna beam steering to improve performance of vehicular Internet access. In *MobiSys* (June 2007).

[84] Nicholson, Anthony J., and Noble, Brian D. Breadcrumbs: forecasting mobile connectivity. In *MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking* (New York, NY, USA, 2008), ACM, pp. 46–57.

[85] Ormont, Justin, Walker, Jordan, Banerjee, Suman, Sridharan, Ashwin, Seshadri, Mukund, and Machiraju, Sridhar. A City-wide Vehicular Infrastructure for Wide-area Wireless Experimentation. In *WiNTECH* (September 2008).

[86] Ott, J., and Kutscher, D. Bundling the Web: HTTP over DTN. In *Proc. Workshop on Networking in Public Transport* (August 2006).

[87] Ott, Jorg, and Kutscher, Dirk. Drive-thru Internet: IEEE 802.11b for automobile users. In *INFOCOM* (Mar. 2004).

[88] Padmanabhan, V., and Mogul, J. Using Predictive Prefetching to Improve World Wide Web Latency. In *Proc. ACM Sigcomm* (July 1996), pp. 22–36.

[89] Papadimitriou, Christos. *Computational Complexity*. Addison Wesley, 1994.

[90] Perkins, C. Ad hoc on demand distance vector (aodv) routing, 1997.

[91] Rahmati, Ahmad, and Zhong, Lin. Context-for-wireless: Context-sensitive Energy-efficient Wireless Data Transfer. In *Proc. MobiSys* (June 2007).

[92] Ramani, Ishwar, and Savage, Stefan. Syncscan: Practical fast handoff for 802.11 infrastructure networks. In *INFOCOM* (2005).

[93] Rodriguez, P., Chakravorty, R., Chesterfield, Julian, Pratt, I., and Banerjee, S. MARS: A commuter router infrastructure for the mobile Internet. In *MobiSys* (June 2004).

[94] S. A. Libby Levison, William Thies. Searching the World Wide Web in Low-Connectivity Communities. In *2002 International Symposium on Technology and Society* (June 2002).

[95] Seshan, Srinivasan, Balakrishnan, Hari, and Katz, Randy H. Handoffs in cellular wireless networks: The Daedalus implementation and experience. *Wireless Personal Communications (Kluwer) 4*, 2 (1997).

[96] Seth, A., Kroeker, D., Zaharia, M., Guo, S., and Keshav, S. Low-cost Communication for Rural Internet Kiosks Using Mechanical Backhaul. In *Proc. ACM Mobicom* (September 2006), pp. 334–345.

[97] Shah, R. C., Roy, S., Jain, S., and Brunette, W. Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks. In *Proc. IEEE SNPA* (May 2003), pp. 30–41.

[98] Shih, Eugene, Bahl, Paramvir, and Sinclair, Michael J. Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices. In *Proc. MobiCom* (September 2002).

[99] Small, Tara, and Haas, Zygmunt. Resource and performance tradeoffs in delay-tolerant wireless networks. In *ACM WDTN* (2005).

[100] Sorber, Jacob, Balasubramanian, Aruna, Corner, Mark D., Ennen, Joshua, and Qualls, Carl. Tula:balancing energy for sensing and communication in a perpetual mobile network. Tech. Rep. 09-58, University of Massachusetts Amherst, 2009.

[101] Soroush, Hamed, Banerjee, Nilanjan, Balasubramanian, Aruna, Corner, Mark D., Levine, Brian Neil, and Lynn, Brian. DOME: A Diverse Outdoor Mobile Testbed. In *Proc. ACM Intl. Workshop on Hot Topics of Planet-Scale Mobility Measurements (HotPlanet)* (June 2009).

[102] Spyropoulos, Thrasyvoulos, Psounis, Konstantinos, and Raghavendra, Cauligi S. Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks. In *Proc. ACM WDTN* (Aug. 2005), pp. 252–259.

[103] Spyropoulos, Thrasyvoulos, Psounis, Konstantinos, and Raghavendra, Cauligi S. Performance analysis of mobility-assisted routing. In *ACM MobiHoc* (May 2006), pp. 49–60.

[104] Srinivasan, Kannan, Kazandjieva, Maria A., Agarwal, Saatvik, and Levis, Philip. The Beta-factor: Measuring Wireless Link Burstiness. In *Proc. SenSys* (October 2008).

[105] Strohman, Trevor, Metzler, Donald, Turtle, Howard, and Croft, W. Bruce. Indri: A Language Model-Based Search Engine for Complex Queries. In *Proc. Intl. Conf. on Intelligence Analysis* (May 2005).

[106] Svennson, Peter. AT&T: Tighter Control of Cell Data Usage Ahead. `http://seattletimes.nwsource.com/html/businesstechnology/2010461891_apustecattdatausage.html`, 2009.

[107] T. Spyropoulos and K. Psounis and C. Raghavendra. Single-copy Routing in Intermittently Connected Mobile Networks. In *IEEE SECON* (October 2004).

[108] Thiagarajan, Arvind, Ravindranath, Lenin, LaCurts, Katrina, Madden, Samuel, Balakrishnan, Hari, Toledo, Sivan, and Eriksson, Jakob. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *SenSys '09: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems* (New York, NY, USA, 2009), ACM, pp. 85–98.

[109] Tie, Xiaozheng, Balasubramanian, Aruna, Somasundaram, Manikandan, and Venkataramani, Arun. Routing for diverse wireless networks. Um-cs-2010, University of Massachusetts Amherst, 2010.

[110] Vanlan. `http://research.microsoft.com/netres/projects/vanlan//`.

[111] Viterbi, Andrew J., Viterbi, Audrey M., Gilhousen, Klein S., and Zehavi, Ephraim. Soft handoff extends CDMA cell coverage and increases reverse cell capacity. *IEEE JSAC 12*, 8 (Oct. 1994).

[112] Widmer, Jorg, and Le Boudec, Jean-Yves. Network Coding for Efficient Communication in Extreme Networks. In *Proc. ACM WDTN* (Aug. 2005), pp. 284–291.

[113] City-wide Wi-Fi rolls out in UK. `http://news.bbc.co.uk/2/hi/technology/4578114.stm`.

[114] Cities unleash free Wi-Fi. `http://www.wired.com/gadgets/wireless/news/2005/10/68999`.

[115] Wortham, Jenna. Customers Angered as iPhones Overload 3G. `http://www.nytimes.com/2009/09/03/technology/companies/03att.html?_r=2&partner=MYWAY&ei=5065/`, 2009.

[116] Xie, Yinglian, and O'Hallaron, David R. Locality in Search Engine Queries and Its Implications for Caching. In *Proc. IEEE Infocom* (June 2002), pp. 1238–1247.

[117] Zhang, Pei, Sadler, Christopher M., Lyon, Stephen A., and Martonosi, Margaret. Hardware Design Experiences in ZebraNet. In *Proc. ACM SenSys* (Nov. 2004), pp. 227–238.

[118] Zhang, X., Neglia, G., Kurose, J., and Towsley, D. Performance Modeling of Epidemic Routing. In *Proc. IFIP Networking* (May 2006).

[119] Zhang, Xiaolan, Kurose, Jim, Levine, Brian Neil, Towsley, Don, and Zhang, Honggang. Study of a Bus-Based Disruption Tolerant Network: Mobility Modeling and Impact on Routing. In *Proc. ACM Intl. Conf. on Mobile Computing and Networking (Mobicom)* (September 2007), pp. 195–206.

[120] Zhou, Yun, Levine, Brian Neil, and Croft, W. Bruce. Distributed Information Retrieval For Disruption-Tolerant Mobile Networks. CIIR Technical Report IR-412, University of Massachusetts Amherst, 2005.

[121] Ziegler, Chris. Sprint Falls in Line, Caps "Unlimited" Data at 5GB. http://www.engadgetmobile.com/2008/05/19/sprint-falls-in-line-caps-unlimited-data-at-5gb/, 2008.