The Dissertation Committee for Upendra Bhalchandra Shevade
certifies that this is the approved version of the following dissertation:

Improving Performance and Incentives in Disruption-tolerant Networks

Committee:

Yin Zhang, Supervisor

James Browne

Aloysius Mok

Lili Qiu

Harrick Vin

Improving Performance and Incentives in Disruption-tolerant Networks

by

Upendra Bhalchandra Shevade, B. E., M.A.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2010

To Baba, Aai, Anu, Nilesh and Siana

||Shrivitthalacharanarpanamastu||

Acknowledgments

The acknowledgment section presents a wonderful opportunity to thank all those who have made both research and life such a joy.

I would like to thank my advisors Prof. Yin Zhang and Prof. Lili Qiu for being such amazing advisors. Their insight, dedication, work ethic and general brilliance will inspire me throughout my life. They have made me both a better computer scientist as well as a better person.

Prof. Harrick Vin supervised me during my Masters and also served on my PhD committee. His acuity and grasp of fundamentals always amaze me. I thank him for his constant support through these years.

I am grateful to Prof. Browne and Prof. Mok for agreeing to be on my PhD committee, for challenging me with pertinent questions, and for providing me valuable feedback.

Over the past year, I had the wonderful opportunity to serve as the teaching assistant for the Computer Architecture course. I thank Prof. Hunt for his rigor, his kindness, and for making the course so much fun. Working with Prof. McKinley was an amazing experience. For teaching me the value of discipline, for giving me the opportunity to teach, and for her advice on both research and life, I am deeply grateful. I must also thank the other TAs for the course–David Rager and Renée St. Amant–for making it all so much fun.

The Laboratory for Advanced Systems Research (LASR) is a crucible of brilliant and dedicated computer scientists. I am proud to have been its member. In addition to Lili and Yin, I thank the other professors in the lab–Prof. Alvisi, Prof. Dahlin, Prof. Walfish and Prof. Witchel–for making the lab the powerful intellectual institution it is today.

Over the seven years I spent as a member of LASR, Sara Strandtman has been the bedrock of our lab. For all the help, big and small, and for her consistent and unflinching support and friendship, I am deeply and truly grateful.

I thank the graduate coordinators–Lydia Griffith, Gloria Ramirez and Katherine Utz–for all the kindness and help through the years.

Special thanks are due to my student co-authors, for my dissertation could not have been completed without their help: Han Hee Song, Mi Kyung Han, Vinoth Chandar, Yi Chao Chen and YouSuk Seung. Thanks Lasrians, past and current–Amitanand Aiyer, Nalini Belaramani, Apurv Bhartia, Tae-Won Cho, Allen Clement, Vacha Dave, Wei Dong, Manos Kapritsos, Ravi Kokku, Rama Kotla, Harry Li, Prince Mahajan, Ajay Mahimkar, JP Martin, Yogita Mehta, Jayaram Mudigonda, Don Porter, Swati Rallapalli, Taylor Riché, Chris Rossbach, Indrajit Roy, Eric Rozner, Jayesh Seshadri, Mitul Tiwari, Edmund Wong, Praveen Yalagandula, and Jiandan Zheng–for all the interesting discussions and the wonderful times.

I must thank all my friends in Austin–Ajay, Amit, Han, Jasraj, Manasi,

Mi Kyung, Mukta, Neeraj, Neha, Nikhil, Nupur, Saurabh, Siddhika, Smriti, TaeWon, Vasanth and Vishwas–for always being there.

I dedicate this thesis to my family: Aai, Baba, Anu, Nilesh and Siana. None of this would have been possible without their love and support.

Improving Performance and Incentives in Disruption-tolerant Networks

Publication No. _____

Upendra Bhalchandra Shevade, Ph.D.
The University of Texas at Austin, 2010

Supervisor: Yin Zhang

The recent proliferation of personal wireless devices has led to the emergence of disruption-tolerant networks (DTNs), which are characterized by intermittent connectivity among some or all participating nodes and a consequent lack of contemporaneous end-to-end paths between the source and consumer of information. However, the success of DTNs as a communication paradigm is critically dependent on the following challenges being addressed: (1) How to enable popular but demanding applications, such as video-on-demand, to operate in such constrained network settings, and (2) How to incentivize individual devices to cooperate when network operation is only possible under, or greatly benefits from cooperation.

In this dissertation, we present a novel set of protocols and develop real systems that effectively meet the above challenges. We make the following contributions:

First, we design and implement a novel system for enabling high bandwidth content distribution in vehicular DTNs by leveraging infrastructure access points (APs). We predict which APs will soon be visited by a vehicular node and then proactively push content-of-interest to those APs. Our replication schemes optimize content delivery by exploiting Internet connectivity, local wireless connectivity, node relay connectivity and mesh connectivity among APs. We demonstrate the effectiveness of our system through trace-driven simulation and Emulab emulation using real taxi and bus traces. We further deploy our system in two vehicular networks: a fourteen AP 802.11b network and a four AP 802.11n network with smartphones and laptops as clients.

Second, we propose an incentive-aware routing protocol for DTNs. In DTNs, routing takes place in a store-and-forward fashion with the help of relay nodes. If the nodes in a DTN are controlled by rational entities, such as people or organizations, the nodes can be expected to behave selfishly by attempting to maximize their utilities and conserve their resources. Since routing is inherently a cooperative activity, system operation will be critically impaired unless cooperation is incentivized. We propose the use of pair-wise tit-for-tat (TFT) as a simple, robust and practical incentive mechanism for DTNs. We then develop an incentive-aware routing protocol that allows selfish nodes to maximize their own performance while conforming to TFT constraints.

Table of Contents

Chapter 1

Introduction

Personal mobile computing devices such as laptops and smartphones have over the past decade replaced the traditional desktop as the premier communication and entertainment device for a majority of users. The trend towards miniaturization and mobility continues with an estimated 1.2 billion internet-capable mobile handsets in use by the end of 2010, and smartphone sales surpassing those of all other categories of computing devices [32]. These mobile devices enable on-the-move communication in ways that were hitherto uncommon.

Disruption-tolerant networks (DTNs) are a communication paradigm that attempt to leverage intermittent communication opportunities available to such mobile devices to provide users with a richer network experience. DTNs are distinguished from other types of networks by the lack of contemporaneous end-to-end path between the source and consumer of information. This characteristic forces communication to take place in a store-and-forward fashion, with intermediate nodes forwarding data opportunistically over fleeting contacts until it reaches the destination.

For DTNs to be successfully adopted as the communication paradigm

for personal mobile devices, two significant challenges, among others, must be addressed. First, users expect similar communication and entertainment experience to be available on mobile devices as the one they are familiar with on the desktop. This requires the resource-challenged network to be able to support demanding applications such as video-on-demand and TV. Second, mobile devices are owned by rational individuals, who must be incentivized to cooperate so that the networks delivers its full potential.

In this dissertation, we build real systems based on a novel set of protocols to address the above challenges. In doing so, we make the following contributions:

1. We design and implement a novel system for high-bandwidth content distribution in DTNs, aimed particularly at vehicular DTNs. To do so, we leverage infrastructure access points (APs) that a vehicular node opportunistically contacts to download content of interest. Given the transient nature of the contact imposed by vehicular speeds, we first predict which APs will soon be visited by a node and then proactively push content-of-interest to those APs. Our replication schemes optimize content delivery by exploiting Internet connectivity, local wireless connectivity, node relay connectivity and mesh connectivity among APs. We demonstrate the effectiveness of our system through trace-driven simulation and Emulab-based emulation using real taxi and bus mobility traces. Further, we deploy our system in two campus-based vehicular networks:

a fourteen AP 802.11b network and a four AP 802.11n network with Windows smartphones and laptops acting as clients.

2. We propose an incentive-aware routing protocol for DTNs. Routing in DTNs takes place in a store-and-forward fashion with the help of relay nodes. If the nodes in a DTN are controlled by rational entities, such as people or organizations, the nodes can be expected to behave selfishly by attempting to maximize their utilities and conserve their resources. Since routing is an inherently cooperative activity, system operation will be critically impaired unless cooperation is somehow incentivized. We propose the use of pair-wise tit-for-tat (TFT) as a simple, robust and practical incentive mechanism for DTNs. We then develop an incentive-aware routing protocol that allows selfish nodes to maximize their own performance while conforming to TFT constraints.

1.1 Challenges

Disruption-tolerant networks of mobile devices create both new research challenges and opportunities:

- **Intermittent connectivity.** Contacts between nodes in a DTN are transient. For example, in a vehicular DTN, 70% of contact opportunities between devices and APs were found to be less than 10 seconds [18]. If meaningful communication is to happen during this time, the contact must be anticipated and the contact time maximally utilized.

- **Limited Internet bandwidth.** While devices communicate wirelessly with each other and with the APs, the connection between the Internet and the APs is through a wireline technology (*e.g.* cable or DSL). Wireless capacity available through WiFi is often an order of magnitude higher than typical Internet connectivity. Hence, it is the Internet connection that becomes the bottleneck during transient contacts.

- **Difficult-to-predict contacts.** Since contacts among various entities in a DTN are dictated by human or vehicular mobility, they are difficult to predict. However, network operation can greatly benefit from prediction of contacts. Mobility history and various constraints on mobility (*e.g.* road structure in case of vehicular DTNs) can be exploited to develop better prediction algorithms.

- **Rational behavior.** Since each mobile device is potentially owned by a different individual, nodes can be expected to behave rationally when participating in system activities such as routing. The protocols devised must hence be incentive-aware if the system is to achieve its communication potential.

- **Limited visibility of actions.** In a DTN, nodes meet and then potentially move out of communication range before they meet other nodes. This means that no node or group of nodes has enough visibility of the network to be able to directly verify a device's claim to have performed an action. The incentive mechanism for DTNs must take this limited

visibility into consideration.

- **WiFi versus 3G.** While cellular technologies such as 3G offer always-on connectivity to the Internet, they have limited bandwidth and incur high cost. While the wireless bandwidth advertised is high, it is the Internet backhaul of the cellular carrier that is usually a bottleneck. In addition, most cellular service providers in US, like AT&T, T-mobile, Sprint, Verizon, charge $60/month for 5GB data transfer and $0.2/MB afterwards. 5GB data transfer can only support 0.1 Mbps for 111 hours (< 5 days)! The cellular service price in many other countries are similar or even higher [74]. Moreover, many mobile broadband providers restrict or limit large data exchanges, including streaming audio, video, P2P file sharing, JPEG uploads, VoIP and automated feeds [50]. According to the international poll of 2700 Devicescape customers, 81% smartphone users prefer WiFi over 3G cellular for data services [57]. Therefore there is strong need for supporting high-bandwidth applications in vehicular networks using WiFi alone, which is widely deployed and typically free.

1.2 Approach

We overcome the above challenges and exploit the unique opportunities offered by DTNs in the following ways:

1.2.1 Enabling high-bandwidth content distribution

We design and implement VCD, a novel system for enabling high-bandwidth content distribution in vehicular DTNs. We assume infrastructure in the form of road-side access points (APs) some of which are connected to the Internet. To fully take advantage of the transient contact between APs and devices, we proactively push content to APs that the vehicles will likely visit in the near future. In this way, vehicles can enjoy the full wireless capacity instead of being bottlenecked by the Internet connectivity, which is either slow or not even available.

To realize this vision, we first develop a new algorithm for predicting the APs that will soon be visited by the vehicles with high accuracy. Given the high driving speeds, diverse and unpredictable road conditions, infrequent location updates, and irregular update intervals, accurately predicting mobility is challenging in vehicular networks. We develop a new mobility prediction algorithm based on the idea of *voting among K nearest trajectories (KNT)*. We also implement several state-of-the-art mobility prediction algorithms based on Markov mobility models [65, 52]. Our evaluation shows that KNT achieves better prediction accuracy on our dataset.

We then develop wireline and wireless replication techniques to leverage the synergy among (i) Internet connectivity (which is persistent but has limited coverage and low bandwidth), (ii) local wireless connectivity (which has high bandwidth but short contact duration), (iii) vehicular relay connectivity (which has high bandwidth but high delay), and (iv) mesh connectivity

among APs, which is persistent and has high bandwidth but low coverage.

We conduct extensive trace-driven simulations to evaluate the performance of VCD using real mobility traces of 500 taxis in the San Francisco Bay Area over the course of 30 days and of 1200 city buses in Seattle over a week. Our results show that VCD is capable of downloading 3-6X as much content as no replication and 2-4X as much content as wireline or wireless replication alone; mesh replication further helps to improve throughput by up to 22%. The benefit of VCD further increases as the ratio between wireless and wireline capacity increases and AP deployment becomes denser.

Encouraged by our simulation results, we developed a full-fledged prototype VCD system that supports real video streaming applications running on smartphones. We deploy our system on two campus-based testbeds: one consisting of fourteen 802.11b APs and another of four 802.11n APs with three smartphones and two laptops acting as clients. Live road tests suggest that our system is capable of providing video streaming to smartphone clients at vehicular speeds. To further evaluate the performance of VCD at scale, we run the same AP and controller code as in the testbed together with emulated vehicles in the Emulab [29] testbed. Our experiments show that Emulab results closely follow the simulator results and that our implementation is efficient and light-weight.

1.2.2 Incentive-aware routing in DTNs

We design an incentive-aware routing protocol for DTNs that provides a basis for devices to reconcile rational behavior with cooperative routing with other devices.

We first study the impact of selfish behavior on DTN operation. A selfish user may drop others' messages and excessively replicate its own messages to increase its own delivery rate while significantly degrading other users' performance or even causing starvation. Using simulation based on both synthetic and real mobility traces, we show that the presence of selfish users can degrade total delivered traffic to less than 20% as what can be delivered under full cooperation. Since DTNs have limited connectivity, simply removing selfish nodes results in serious performance penalty.

In the absence of an incentive mechanism, it is rational for nodes to behave selfishly. Hence, an incentive mechanism is needed to provide a basis for cooperation among devices. We note that given the limited visibility of a node's actions to any other node in the system, we cannot reliably detect if a node misbehaved and did not forward traffic for another node. This implies that we cannot use the detect-and-punish approach [45], successfully applied for MANETs, in DTNs.

Instead, we propose the use of pairwise tit-for-tat (TFT) as a simple, robust, and practical incentive mechanism for DTNs. TFT gracefully deals with limited visibility by not attempting to detect bad behavior, but

instead, by rewarding only good behavior *i.e.* correct forwarding. However, existing TFT mechanisms often face bootstrapping problems or suffer from exploitation. We propose a TFT mechanism that incorporates generosity and contrition to address these issues.

We then develop an incentive-aware routing protocol that allows selfish nodes to maximize their individual utilities while conforming to TFT constraints. We also address the practical challenges involved in implementing the TFT mechanism. We evaluate the effectiveness of our incentive-aware routing scheme using both synthetic and real DTN traces. Our results show that with TFT as a basis of cooperation among selfish nodes, the total delivered traffic increases to 60% or higher as under full cooperation.

Adoption of the disruption tolerant paradigm: The disruption-tolerant communication paradigm has been proposed for use in a variety of contexts: to build tactical military networks [27, 28], to quickly reestablish connectivity after natural disasters [31], to provide basic connectivity in underdeveloped regions [63], for wildlife tracking [39] and as a viable paradigm for underwater [54] and inter-planetary communication [12]. Common to all of these scenarios is the absence or unreliability of ubiquitous connectivity that modern cellular 3G and 4G networks provide in urban areas of developed countries. However, as our VCD system demonstrates, DTNs built with infrastructure WiFi support can be used to provide high-bandwidth content transfers, and can hence act as a viable alternative or supplement to the cellular networks. This is particularly germane today as limited spectrum, backhaul limitations

and dramatic increase in bandwidth usage has lead to poor performance for many cellular networks. The industry has already taken the first steps towards standardizing the mechanisms to intelligently offload content transfers to WiFi [59]. Further real world adoption of DTNs depends on how security and incentive issues are addressed, and whether certain applications that need or can work with DTNs become popular.

The remainder of the dissertation is organized as follows. We present our work on enabling high-bandwidth content distribution in Chapter 2. Chapter 3 discusses incentive-aware routing in DTNs. Finally, we conclude and discuss future research avenues in Chapter 4.

Chapter 2

High-bandwidth Vehicular Content Distribution

Vehicular networks have emerged from the strong desire to communicate on the move [6, 7, 30, 71]. Car manufacturers all over the world are developing industry standards and prototypes for vehicular networks (*e.g.*, [9, 17, 68]). Existing works on vehicular networks often focus on low-bandwidth applications, such as credit card payment, traffic condition monitoring [18], Web browsing [6, 7], and VoIP [7]. We explore the potential of supporting high-bandwidth applications (*e.g.*, video streaming) in vehicular networks.

Challenges and opportunities:

A natural way for vehicular clients to download content from contacts with roadside APs, is to request and download data from the Internet via the WiFi connection, as and when the contact happens [7, 30]. However, it is challenging to meet high bandwidth requirement since vehicles often move at a high speed and thus the contact time between vehicles and APs tends to be short (*e.g.*, [18] reported that 70% of connection opportunities are less than 10 seconds). In addition, it is often expensive to provide dense high-speed WiFi-based Internet coverage at a large scale. As a result, if vehicles

fetch desired content on-demand from the Internet during their contact with an AP, the amount of data fetched may be insufficient to sustain the data rate required by applications such as video streaming when vehicles are outside the communication range of any APs.

With recent advances in wireless technology, WiFi capacity has grown rapidly and can be an order of magnitude higher than typical Internet access link connectivity. For validation, we performed a measurement experiment using a laptop equipped with NetGear WNDA3100 on a vehicle communicating with a NetGear WNDR3300 AP deployed near the road. We obtained 4.6Mbps using 802.11b, 22.2Mbps using 802.11g, and 39.7Mbps using 802.11n on 2.4GHz frequency, and 56.1Mbps using 802.11n on 5GHz. In comparison, DSL throughput ranges between 768Kbps to 6Mbps [3], which is an order of magnitude slower. The gap between the wireline and wireless capacity is likely to increase further (*e.g.*, due to the availability of new spectrum, such as whitespace, and advances in antenna and signal processing technology). Such large gap suggests that in order to enjoy high wireless capacity, we should proactively replicate content beforehand to the APs that a vehicle is likely to visit. While the idea of replication is natural, *how to replicate the content given the limited wireline and wireless resources and uncertainty in vehicular trajectory* is an open research question that we aim to address.

Approach and contributions: We develop a replication strategy that effectively exploits the synergy among (i) Internet connectivity, which is persistent but has limited coverage and relatively low bandwidth, (ii) local wireless

connectivity, which has high bandwidth but short contact duration, (iii) vehicle relay connectivity, which has high bandwidth but high delay, and (iv) mesh connectivity among APs, which is persistent and has high bandwidth but low coverage. In particular, we optimize replication through wireline network and wireless mesh networks based on predicted mobility and traffic demands. Moreover, we opportunistically exploit the mobility of the vehicles to extend the coverage of the Internet and mesh connectivity. Even if only a small fraction of APs have Internet and mesh connectivity, by having the vehicles themselves relay content, one can potentially replicate content to a much larger number of APs. In essence, vehicle mobility has the potential to significantly increase the network capacity [34] and reduce future content access delay. Note that many mobile devices, such as smartphones, support the use of cheap external storage cards, which can help mitigate potential storage concerns regarding carrying traffic for other users in the system [67].

To this end, we develop a novel **V**ehicular **C**ontent **D**istribution (VCD) system for enabling high-bandwidth content distribution in vehicular networks. As illustrated in Figure 2.1, VCD consists of vehicles, APs with and without Internet access (some of which may form a mesh network), content server on the Internet (*e.g.*, Web servers), and a controller. Vehicles submit location updates and content requests to the controller via cellular links. The controller optimizes the replication strategy based on predicted mobility and traffic demands, and instructs the APs to carry out the replication strategy. To enhance reliability and scalability, the controller can be replicated on multiple nodes.

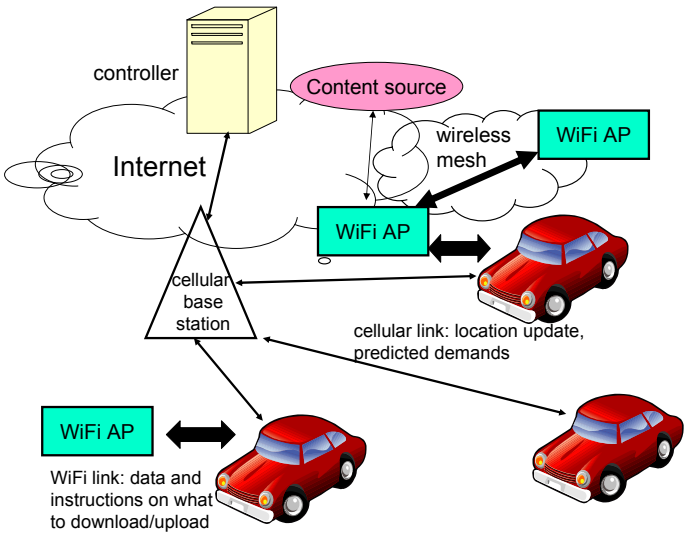


Figure 2.1: VCD architecture

APs are deployed along road sides (*e.g.*, at gas stations and/or coffee shops) to allow vehicles on the road to opportunistically communicate with them. The APs prefetch content as instructed by the controller. Whenever a vehicle encounters an AP, the AP tries to send the requested content from its local storage if the content is available locally. Otherwise, the AP tries to fetch the content from an AP in the same mesh network if one is available. If no such AP is found, it fetches content from the Internet when it has Internet connectivity. In addition to sending the content that the vehicle itself needs, the AP may also send the vehicle content that can then be relayed to other APs, or fetch from the vehicle content that can be served to other passing vehicles later.

VCD systems are easy to deploy in practical settings. For example, a vehicular service provider (VSP) can install its own APs and/or subscribe to existing wireless hotspot services. Since it is easy to place a stand-alone AP than hooking it up with Internet connection, VCD is designed to explicitly take advantage of APs with and without Internet connectivity. An AP without Internet connectivity is still useful since it can serve as a static cache, which vehicles can upload content that can be served to other passing vehicles in the future.

VSPs can offer content distribution service to taxis, buses, subways, and personal vehicles. We focus on taxis and buses that offer high-bandwidth content distribution as a value added service to their passengers. These vehicles have low-cost mobile devices on board for playing downloaded content. Such mobile devices can be installed by either the taxi/bus companies or VSPs. Since the mobile devices can be powered by the vehicles, power consumption is not an issue. The mobile devices interact with APs and the VCD controller to report required information (*e.g.*, location update and predicted traffic demands) and follow their instructions.

The key contributions of VCD include:

1. *Optimized wireline and mesh replication.* To fully take advantage of short contact time between APs and vehicles, we replicate content in advance to the APs that will soon be visited by the vehicle. A distinctive feature of our replication scheme is that it is based on optimization. Specifically,

we explicitly formulate a linear program (LP) to optimize the amount of data that can be delivered before the deadline under the predicted mobility pattern and traffic demands subject to given resource constraints (*e.g.*, short contact time and limited link capacity). In contrast, previous works either focus exclusively on protocol-level optimization of one-hop communication between vehicles and APs (*e.g.*, [18, 7, 16, 51]), or rely on heuristics to guide data replication [20], or completely ignore the resource constraints [26], which are crucial in vehicular networks. Our formulation is highly flexible and can be adapted to support both wireline replication (Section 2.1.2) and mesh replication (Section 2.1.3). The formulation can be efficiently solved using standard LP solvers such as *lp_solve* [44] and *cplex* [24], owing to modern interior-point linear programming methods.

2. *Opportunistic vehicular replication.* To further extend the coverage of the Internet and wireless mesh networks, we develop vehicular replication to opportunistically take advantage of local wireless connectivity and vehicular relay connectivity (Section 2.1.4). Our scheme explicitly leverages the APs as the rendezvous points for replicating data among vehicles. Such an approach is more effective than the traditional approach of direct vehicle-to-vehicle replication according to the theoretical analysis of [8].

3. *A new algorithm for mobility prediction.* For our replication optimization algorithms to be effective, it is critical to predict the set of APs a vehicle

will visit in a future interval with high accuracy. Given the high driving speeds, diverse and unpredictable road conditions, infrequent location updates, and irregular update intervals, accurately predicting mobility is challenging in vehicular networks. We develop a new mobility prediction algorithm based on the idea of *voting among K nearest trajectories (KNT)* (Section 2.2). We also implement several state-of-the-art mobility prediction algorithms based on Markov mobility models [65, 52]. Our evaluation in Section 2.4 shows that *KNT* achieves better prediction accuracy on our dataset.

4. *Thorough evaluation through simulation, emulation, and testbed experiments.* We conduct trace-driven simulations to evaluate the performance of VCD using San Francisco taxi [15] and Seattle bus traces [61] (Section 2.5). Our results show that VCD is capable of downloading 3-6X as much content as no replication, and 2-4X as much content as wireline or vehicular replication alone; mesh replication further helps to improve throughput by up to 22%. The benefit of VCD further increases as the gap between wireless and wireline capacity enlarges and the AP density increases. In addition, we have developed a full-fledged prototype VCD system that supports real video streaming applications running on smartphones and laptops (Section 2.3, 2.6 and 2.7). We deploy our system in two wireless testbeds using 802.11b and 802.11n. Live road tests suggest that our system is capable of providing video streaming to smartphone and laptop clients at a vehicular speed. To further evaluate the

performance of VCD at scale, we run the same AP and controller code as in the testbed together with emulated vehicles in the Emulab [29]. Our experiments show that Emulab results closely follow the simulator results and our implementation is efficient.

Paper organization: The remainder of this chapter is organized as follows. We present our replication optimization techniques in Section 2.1 and our mobility prediction algorithm in Section 2.2. We specify our system design and implementation in Section 2.3. We evaluate the accuracy of mobility prediction in Section 2.4. We evaluate the performance of VCD through trace-driven simulation in Section 2.5, trace-driven emulation in Section 2.6, and testbed experiments in Section 2.7. We survey related work in Section 2.8.

2.1 Optimizing Replication

In this section, we first present an overview of our system, and then develop wireline, mesh, and vehicular replication.

2.1.1 Overview

At the beginning of every interval, the controller collects the inputs required for computing replication strategy. The controller computes the replication strategy during the current interval so that it can maximize user satisfaction during the next interval (Section 2.1.2). We use user satisfaction in the next interval as the objective since replication in the current interval is

often too late to satisfy the traffic demands in the same interval. The controller then informs the APs of the replication strategy through the Internet or cellular network (in case the APs do not have Internet connectivity). We use cellular networks to send control messages as they are small. When a vehicle contacts an AP, the AP transfers the content according to the optimization results (step 1). When this finishes, the vehicle may still have demand that has not yet been satisfied (*e.g.*, due to inaccurate prediction or insufficient capacity to replicate all the interesting content). The vehicle will first download all the content that the vehicle is interested in and also available locally at the AP (step 2). Then it downloads the remaining content that it is interested in from the AP's mesh network or the wireline network when the AP has wireline connectivity (step 3). Parallel to the Internet download (step 4), the vehicle can take advantage of wireless capacity by opportunistically transferring files to and from APs (Section 2.1.4).

2.1.2 Optimized Wireline Replication

Problem formulation: Our goal is to find a replication strategy that maximizes user satisfaction subject to the available network capacity. Specifically, we want to determine how to replicate files to APs during the current interval to maximize the amount of useful content that can be downloaded by vehicles when vehicles meet the APs in the next interval. To support delay sensitive applications, only content that are downloaded before the deadline counts and the other content that already misses the deadline will be excluded from

consideration for replication. This replication problem involves answering the following questions: (i) in what form to replicate the content, and (i) how much to replicate for each file.

Applying network coding: To answer the first question, we note that directly replicating original content introduces two major problems. *First*, it is inefficient for serving multiple vehicles. Suppose multiple vehicles are interested in the same file and have downloaded different portions of the files before their contacts with an AP. If they visit the same AP, in order to satisfy all vehicles we need to replicate the union of the packets they need, which is inefficient. For example, vehicles 1 and 2 are both interested in file 1. Vehicle 1 has downloaded the first half and vehicle 2 has downloaded the second half before they encounter the AP. We need to replicate the complete file to satisfy both vehicles. *Second*, replicating original files is also unreliable. Consider a vehicle is expected to visit three APs but in fact it only visits two of the three APs, which is quite common due to prediction errors. If we just split the file into three and transfer one part to each AP, then the vehicle will not get the complete file. However, if we split the files into two and transfer one part to each AP, the vehicle still may not get the complete file since it may get two redundant pieces (*e.g.*, when it visits the two APs that both have the first half of the file).

We apply network coding to solve both problems. Specifically, we divide the original content into one or multiple files, each containing multiple packets. We use random linear coding to generate random linear combinations

of packets within a file. With a sufficiently large finite field, the likelihood of generating linearly independent packets is very high [36]. For a file with n packets, a vehicle can decode it as long as it receives n linearly independent packets for it.

Network coding solves redundancy problems in multi-vehicle case since each linearly independent packet adds value. In the above example of two vehicles, we only need to replicate one half worth of file content to satisfy both users, reducing bandwidth consumption by half. It solves reliability issue in the single vehicle case by incorporating redundancy. In the above example, we can split the file of interest into 2 and randomly generate 3 linear combinations of these 2 pieces and replicate one to each AP. Since any two pieces are linearly independent with a high probability, the vehicle can decode the file once it gets any two pieces. To avoid redundancy in wireline, mesh and vehicular replication, network coding is performed at content servers, APs, and vehicles. In Section 2.3.2, we describe network coding cost and optimization.

Optimizing replication traffic: Using network coding, we transform the original problem of determining which packets to replicate into the problem of determining how much to replicate for each file. To solve the latter problem, we formulate a linear program, as shown in Figure 2.2. A few explanations follow. The first term in the objective function, $\sum_v \sum_f \sum_{a \in AP(v)} Q(v,f)D(v,f,a)$, quantifies user satisfaction, which is essentially the total traffic downloaded by a vehicle, denoted as $D(v,f,a)$, weighted by the probability for vehicle v to be interested in file f, denoted by $Q(v,f)$. The second term in the objective

$\triangleright$ *Input* : $Intv, WCap, InCap, OutCap, CT, AP, size, has, Q$

$\triangleright$ *Output* : $x(f,i,a)$ *and* $D(v,f,a)$

Maximize: $\sum_{v}\sum_{f}\sum_{a\in AP(v)} Q(v,f)D(v,f,a) - \gamma \sum_{i\in I}\sum_{a\in A}\sum_{f} x(f,i,a)$

Subject to:

[C1] $\sum_{f} D(v,f,a) \leq WCap(a) \times CT(a,v)$ $\quad \forall v, a \in AP(v)$

[C2] $\sum_{a\in AP(v)} D(v,f,a) \leq size(f) - has(v,f)$ $\quad \forall v, f$

[C3] $D(v,f,a) \leq has(a,f) + \sum_{s\in I} x(f,i,a)$ $\quad \forall v, f, a \in AP(v)$

[C4] $\sum_{i\in I} x(f,i,a) \leq size(f) - has(a,f)$ $\quad \forall f, a \in A$

[C5] $\sum_{i\in I}\sum_{f} x(f,i,a) \leq InCap(a) \times Intv$ $\quad \forall a \in A$

[C6] $\sum_{a\in A}\sum_{f} x(f,i,a) \leq OutCap(i) \times Intv$ $\quad \forall i \in I$

Figure 2.2: Optimizing wireline replication, where v is a vehicle, f is a file, a is an AP, i is a node with wireline connectivity (which may or may not be an AP, *e.g.*, a Web server), $Intv$ is an interval duration, A is the set of all the APs, I is the set of all the nodes with wireline connectivity, $AP(v)$ is the set of APs that vehicle v will visit, $Q(v,f)$ is the probability that v is interested in file f, $D(v,f,a)$ is the amount of traffic in file f vehicle v should download from AP a during a contact in the next interval, $x(f,n_1,n_2)$ is the amount of traffic in file f to replicate from node n_1 to node n_2 during the current interval, $CT(a,v)$ is average contact time of vehicle v at AP a, $WCap$ is wireless capacity, $InCap$ is incoming wireline access link capacity, $OutCap$ is outgoing wireline access link capacity, $has(n,f)$ is amount of file f a node n has, and $size(f)$ is the size of file f.

represents the total amount of wireline replication traffic. We include both terms to reflect the goals to (i) maximize user satisfaction, and (ii) prefer the replication that uses less traffic among the replication strategies that support the same amount of traffic demands. Since the first objective is more important, we use a small weighting factor γ for the second term. Our evaluation uses $\gamma = 0.001$.

Constraint C1 in Figure 2.2 enforces that the total amount of traffic downloaded from an AP during a contact is bounded by the product of AP's

wireless capacity and average contact duration. Constraint C2 ensures that the total content downloaded for each file does not exceed the total file size minus the amount of file the vehicle already has before the download. Constraint C3 encodes the fact that the amount of file the vehicle can download from an AP cannot exceed what AP already has plus what will be replicated to the APs through the wireline network during the current interval. Constraint C4 indicates that the total replication traffic in file f towards an AP is bounded by the file size minus the amount that the AP already has. Constraints C5 and C6 reflect the total replication traffic through the wireline network does not exceed the access link capacity. The formulation can support APs with and without wireline access by setting wireline capacity to zeros for APs without wireline access.

Obtaining input: As shown in Figure 2.2, we need $Intv$, $WCap$, $InCap$, $OutCap$, CT, AP, $size$, has, and Q. The $Intv$ is a control parameter that determines how frequently the optimization is performed. In our evaluation, we set $Intv$ to be 3 minutes, which gives a good balance between achieving accurate mobility prediction and limiting the optimization overhead. The next three inputs on link capacity—$WCap$, $InCap$, and $OutCap$—are known in advance and change infrequently. CT is estimated using historical data and only needs to be updated infrequently. For ease of estimation, in our evaluation we set $CT(a, v)$ to be the average duration of all contacts from the trace. AP can be obtained by either letting a vehicle run a mobility prediction algorithm locally or have it send several of its recent GPS coordinates to the controller,

which will perform mobility prediction. *size*, *has*, and Q are reported by the vehicles either through a WiFi link during a contact with an AP or via a cellular link during other time. A vehicle predicts what future content to request based on the previous and current requests. For streaming content, it is relatively easy to predict as most users will request the subsequent frames. Demand prediction in general has been a well-researched problem in many domains [53, 6] and we can leverage existing solutions. Note that all the control information is small and can be easily compressed by sending delta from the previous update.

Using optimization results: *To enhance robustness against errors in estimating the inputs, we use $x(f, i, a)$ and $D(v, f, a)$ to control the relative replication and download rates across different files.* For example, file 1 should be downloaded twice as fast as file 2. In this way, we can still fully utilize network resources even if contact time, wireline and wireless capacity have significant estimation errors.

2.1.3 Optimized Mesh Replication

If some APs along the road are close together, they can form a mesh network. The mesh connectivity indicates that (i) we can now replicate content to the APs using mesh connectivity in addition to wireline connectivity, and (ii) if a vehicle meeting AP1 requests a file that AP1 does not have, it is more efficient to fetch from its mesh network (if there is an AP having the file) than fetching via the slow wireline access link. A neighboring AP in

the mesh network can have the file either due to explicit mesh replication or opportunistically caching from earlier interactions.

To support (i), we make the following modifications to the replication formulation in Figure 2.2. Let $MCap$ denote the capacity of a wireless link in the mesh network, which can be different from the capacity of wireless links between vehicles and APs ($WCap$). Let $z(f, a', a)$ denote the amount of content to replicate from AP a' to a for file f through the mesh network. Let $ETX(a', a)$ denote the average number of transmissions required to send a packet from a' to a through the mesh and can be easily estimated by measuring link loss rate [23]. Our modifications include:

1. adding $-\gamma \sum_f \sum_{(a',a)\in mesh} z(f, a', a)$ to the objective function to prefer the replication that uses less mesh traffic among the ones that support the same traffic demands,

2. adding $+\sum_{(a',a)\in\ mesh} z(f, a', a)$ to the right handside of [C3] to indicate a node can download from AP a any content already available at a or replicated to a through either the wireline or mesh network,

3. adding two new constraints: $z(f, a', a) \leq has(a', f)$ and $\sum_{f,(a',a)\in\ mesh} ETX(a', a) z(f, a', a)/MCap \leq 1$.

The former constraint ensures AP a' cannot replicate more content than it has. The latter is interference constraint, which enforces that total active time of all mesh nodes cannot exceed 100% assuming all nodes in the mesh

network interfere with each other. Note that its left-handside computes activity time by multiplying the replicated content by the number of expected transmissions normalized by the wireless capacity. To support (3), when AP a receiving a request for a file that it does not have locally, it first tries to get from AP a' in the same mesh if the end-to-end throughput (approximated as $MCap/ETX(a', a)$) is higher than wireline access link; only when no such AP is found, does it fetch using the wireline access link.

2.1.4 Opportunistic Vehicular Replication

In addition to wireline and mesh replication, content can also be replicated using vehicles – a vehicle can carry content from one AP to another as it moves. This new form of replication is more effective than traditional vehicle-to-vehicle (V2V) replication, because according to [8] V2V is effective only under a very large number of vehicles whereas a small number of APs can significantly enhance the performance. One way to support this new vehicular replication is to augment the LP formulation in Figure 2.2 with vehicular replication terms, which can produce wireline, mesh and vehicular replication as the final output. However, due to unpredictability in vehicular relay opportunity, we find the effectiveness of such optimization is rather limited. Interestingly, we find the following simple vehicular replication strategy is effective.

Since the wireline fetch is bottlenecked by the slow access link, the wireless link is not fully utilized. Therefore, as mentioned in Section 2.1.1, parallel to the wireline fetch, a vehicle can take advantage of local wireless

connectivity to exchange content with the AP. Such exchange has two benefits: (i) the vehicle can upload content to the AP, which can serve other vehicles later, and (ii) the vehicle can download files, which may serve the user's demand in the future or the vehicle can relay the content to other APs for future service. To enhance effectiveness, we order the files to upload based on the expected future demand for the file at the AP, which is estimated as $\sum_{v:\ v \text{ visits } a} Q(v, f) demand(v, f)$, where $demand(v, f)$ is the expected size of file f vehicle v is interested in. While this vehicular replication is simple, our evaluation shows that it is highly effective.

2.2 Predicting Mobility

If we can predict the AP that a vehicle will visit, we can start replicating the required content to the AP well before the vehicle arrives so that the vehicle can enjoy high wireless bandwidth during its download. Predicting mobility for vehicles is challenging because (i) vehicles often move at high speed, (ii) the GPS updates often have relatively low frequency (*e.g.*, once per minute) and tend to arrive at irregular intervals, and (iii) the road and traffic conditions are highly dynamic and difficult to predict. To address the challenge, we develop a novel mobility prediction algorithm for vehicular networks: *K Nearest Trajectories (KNT)*. We also implement two existing algorithms based on Markov mobility models [65, 52]. In Section 2.4, we show that *KNT* achieves better accuracy on our dataset.

Algorithm: We observe that the mobility of vehicles exhibits unique struc-

ture – a vehicle follows the roads and only makes turns at the street corners or highway exits. This suggests that a good predictor should take into account the speed and direction in the previous interval as well as the underlying road structure. Our *KNT* algorithm is able to account for such information without requiring explicit knowledge about the detailed road map. Given a vehicle v_0 and current time t_0, the algorithm predicts the set of APs visited by v_0 in a future interval $[t_0 + \Delta_1, t_0 + \Delta_2]$ ($\Delta_2 \geq \Delta_1 \geq 0$) in two steps:

1. *Finding K nearest trajectories.* Our algorithm first finds K existing mobility trajectories in a GPS location database that best match the recent mobility history of the given vehicle. Specifically, we maintain a database of past GPS coordinate updates: $\mathcal{D} = \{(v, t, c)\}$, where v is a vehicle, t is the time for the update, and c is the GPS coordinate. For any vehicle v and current time t, we define its mobility history MH as the set of GPS coordinates reported by v in the past δ seconds: $MH_v^t = \{c|(v, s, c) \in \mathcal{D} \wedge s \in [t-\delta, t]\}$. We also define a distance function between two trajectories: $f(MH_{t_0}^{t_0}, MH_v^t) = \sum_{c \in MH_{t_0}^{t_0}} \min_{d \in MH_v^t} \|c - d\|_2$, where $\|c-d\|_2$ is the Euclidean distance between the two locations specified by GPS coordinates c and d. Essentially, this distance function reflects the total distance from each point on $MH_{t_0}^{t_0}$ to the closest point on MH_v^t. We then find K pairs of (v, t) that minimizes $f(MH_{v_0}^{t_0}, MH_v^t)$, *i.e.*, the K nearest neighbors of (v_0, t_0).

2. *Voting.* For each (v, t) among the K nearest neighbors of (v_0, t_0), we use

linear interpolation (*i.e.*, using a line to connect two adjacent points) to obtain its mobility trajectory in its future interval $[t + \Delta_1, t + \Delta_2]$. We can then obtain the set of APs visited by v during this interval. We then report all those APs that are visited by at least T out of K nearest trajectories as the predicted set of APs that will be visited by v_0 during future interval $[t_0 + \Delta_1, t_0 + \Delta_2]$.

In step 1 above, to avoid computing $f(MH_{v_0}^{t_0}, MH_v^t)$ for all pairs of trajectories (which is expensive), we only compute for the trajectory pairs that are nearby. To quickly identify the trajectories that are close to the current one, we create an efficient index structure by (i) discretizing the GPS latitude-longitude coordinate space into $0.0001^\circ \times 0.0001^\circ$ grid squares, and (ii) storing all the (v, t) inside each grid square. Given (v_0, t_0), we start from its grid square and use expanded ring search to find C candidate points (v, t) residing in the same or nearby grid squares. We then find K nearest neighbors among these C candidate points.

Parameter setting: Our algorithm has four control parameters: the number of nearest trajectories K, the number of candidate points C, the voting threshold T, and the mobility history duration H. In our evaluation, we keep $C = 32$, vary $T = 1, 2$, vary K from 2 to 12, and vary H from 60 to 180 seconds. Our results show that $(K = 4, T = 2, C = 32, H = 60)$ consistently give the best performance. We thus only report the results under this parameter setting.

2.3 VCD Implementation

We implement VCD in both Emulab [29] and our real testbed with smartphone and laptop clients. VCD consists of a controller, APs, content servers, and clients in vehicles. Emulab and testbed use the same controller, AP, and content server implementation, all of which are implemented as multi-threaded C++/Linux programs. They differ in client implementation. In Emulab, we implement a virtual vehicle program, which can emulate multiple vehicles, allowing us to conduct a trace driven emulation of all the cars in our trace using a few virtual vehicles. The client in the real testbed is implemented on both smartphones and laptops, which is described in Section 2.3.2.

2.3.1 System Overview

Communication between APs and controller: The APs and controller communicate with each other using TCP. As noted in Section 2.1.1, at the beginning of every interval the controller collects inputs, computes the replication strategy, and instructs content servers or APs to perform wireline and mesh replication at the desirable rates.

Communication between AP and vehicle: The communication between APs and vehicles uses UDP. When a vehicle contacts an AP, it sends a HELLO message that includes (i) a list of ids and sizes of the files it already has, (ii) the files it is interested in during the current and next intervals. Upon receiving the first HELLO message from the vehicle, the AP initiates data download to the vehicle according to the four steps described in Section 2.1.1. Due to the

use of soft state, our protocol works fine when the communication terminates at any time during these steps as the contact is over. Meanwhile, the vehicle also sends buffered GPS updates (generated every 20 seconds in the testbed and every 1 minute in Emulab). Parallel to step 4, the AP determines a list of files for the vehicle to upload sorted in increasing utility as described in Section 2.1.4. The AP sends this list in a REQ message. Upon receiving the first REQ message, the vehicle initiates data upload to the AP. Both HELLO and REQ messages are soft state control messages sent periodically once every control interval (100ms in testbed and 1s in Emulab). These messages also serve as heartbeats to the other party. To achieve efficiency and reliability for data traffic, an AP applies network coding before delivering the data it receives. In addition, to further enhance scalability, we use multiple content servers and leverage a central dispatcher to distribute requests to an appropriate content server for load balancing.

2.3.2 Client Implementation

We implemented client on both Windows XP laptops and smartphones. We use HP Ipaq 910 Business Manager smartphones with Windows Mobile 6.1 Professional operating system, Marvell PXA270 416 MHz Processor, 128MB RAM, Marvell SDIO8661 802.11 b/g WiFi card, and the .Net Compact Framework. Our implementation on smartphones uses OpenNet API, and that on Windows uses Managed WiFi API. Implementing on smartphones introduces several challenges: (i) limited APIs and often inconsistent implementations,

(ii) expensive I/O, (iii) limited system resources, and (iv) many existing wireless optimizations cannot be implemented due to lack of low level access, which we will address.

Handling expensive I/O: Since I/O on smartphones is around an order of magnitude slower than desktops, packets cannot be stored on the disk and read back on-demand for vehicular replication. For simplicity, we use an in-memory packet buffer with FIFO replacement policy. We further limit disk access during the contact with APs and push data to the disk only after the contact is over so that we can fully utilize the short contact time for data transfer.

Handling network coding cost: Due to the slow processor, thread scheduling and dynamic assignment of priorities are important. For example, network coding incurs much higher cost on the smartphone than on the desktop as shown in Table 2.1. We use packet size of 1230 bytes (*i.e.*, the packet payload in our testbed implementation to ensure the maximum packet size is still within 1500 bytes (Ethernet MTU)). Our evaluation uses file sizes of 35, 70, 110 packets, which correspond to minimum, median and maximum file sizes used in our experiments. To minimize the impact of coding cost, we schedule decoding thread at a low priority during a contact and increase the priority when the contact is over.

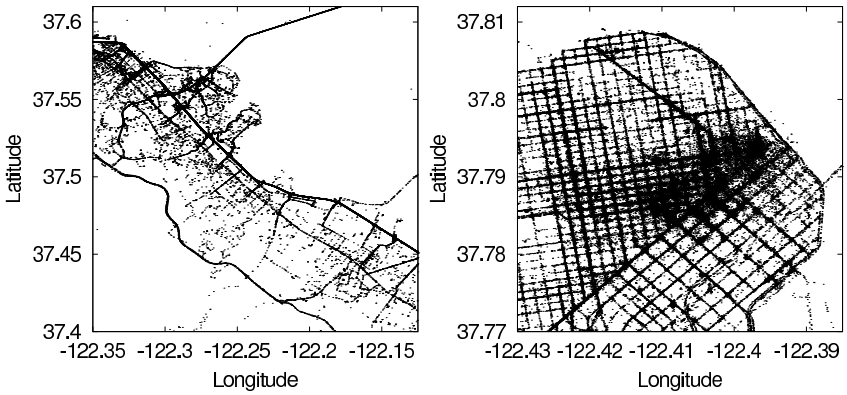
Connection setup: The ability to quickly establish connection to an AP is crucial. [14, 35] examine this problem in greater detail. In the context of

Batch size	110 packets		70 packets		35 packets	
Device	Phone	Desktop	Phone	Desktop	Phone	Desktop
Encoding	12.19s	0.0228s	4.79s	0.0088s	1.18s	0.0021s
Decoding	8.22s	0.017s	3.27s	0.0067s	0.809s	0.0012s

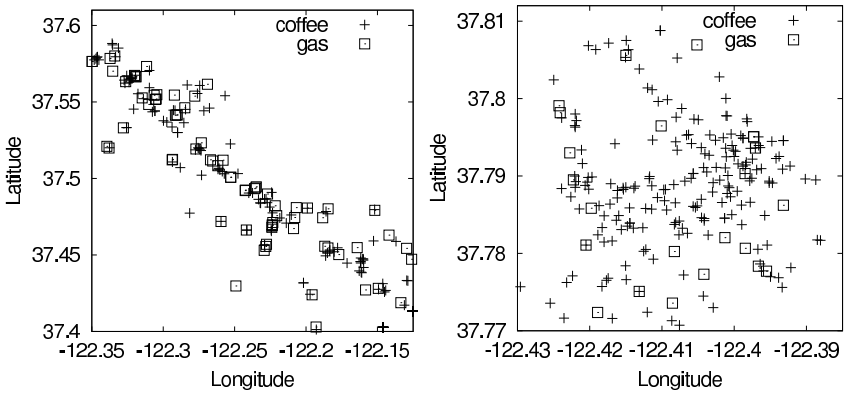
Table 2.1: Network coding benchmarks

smartphones, the problem becomes even harder since NDIS does not provide access to many low level parameters to implement the association optimizations proposed in the literature. Windows Mobile provides two ways to initiate connection to a WiFi network programmatically, either through the wireless zero config (WZC) interface or by setting the appropriate NDIS OIDs. The association times using the WZC interfaces were around 3.0 sec, which is unacceptable in the vehicular network context. We therefore disable WZC and implement NDIS based association, which yields significantly lower association times. We also implement our own DHCP client and use the DHCP caching mechanism described in [14].

Our connection setup procedure is as follows. The smartphone scans for APs every 100 ms. When an AP is discovered, the smartphone waits for 3 RSSI readings greater than -91dB before trying to associate. We do not associate immediately because an association failure is expensive. The association procedure is retried up to 7 times with a short delay of 50ms between consecutive attempts. The various threshold values used in the scheme were chosen empirically. We report the association time and failures in Section 2.7.1.



(a) Taxi locations (on highways) (b) Taxi locations (inside city)



(c) AP locations (on highways) (d) AP locations (inside city)

Figure 2.3: Illustration of traces for mobility prediction.

2.4 Mobility Prediction Accuracy

Mobility traces: We obtain real vehicular mobility traces from Cabspotting [15] and Seattle [61]. The former contain over 10 million GPS longitude and latitude coordinates for approximately 500 taxis in the San Francisco Bay Area over the course of 30 days (December 13, 2008 – January 13, 2009). The latter contains several week-long traces of city buses in Seattle during 2001. The bus system consisted of over 1200 vehicles covering a 5100 square kilometer area.

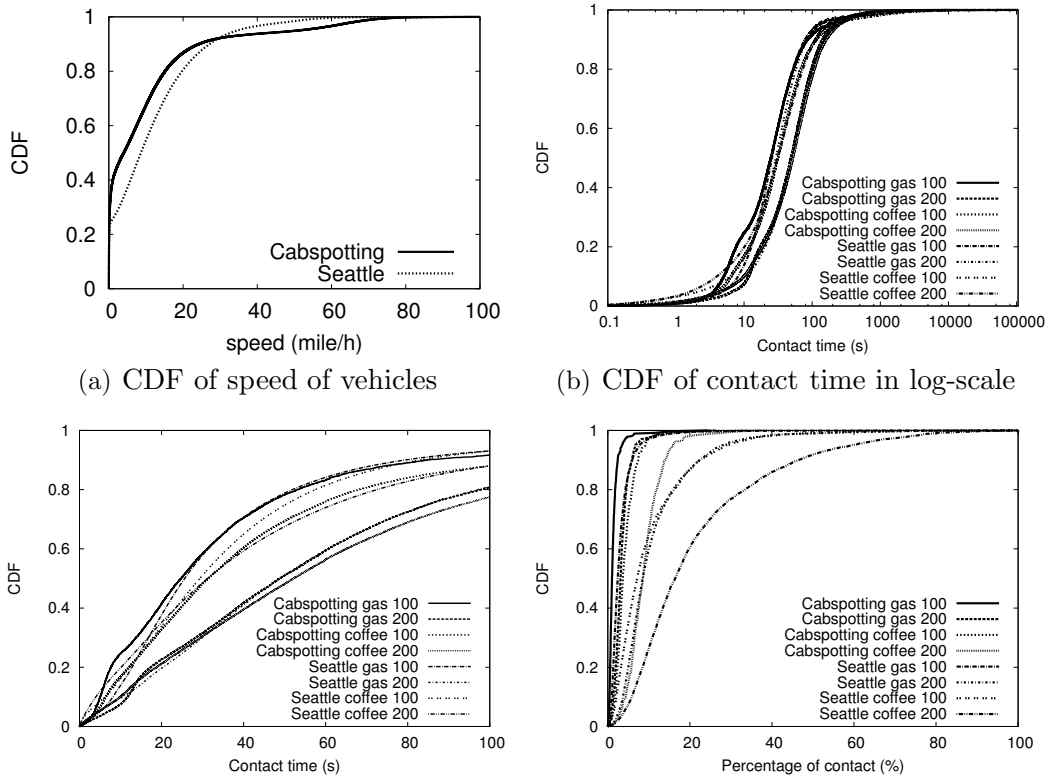
The GPS coordinates are updated approximately once per minute for both Cabspotting and Seattle traces. Figure 2.3 (a) and (b) illustrate the vehicle locations along the highway and inside San Francisco. One can clearly observe the underlying street structure from taxis' GPS. Similar pattern was observed in Seattle traces.

	San Francisco				Seattle			
	median	mean	min	max	median	mean	min	max
Gas	433	589	0.6	7215.1	218	474	1.0	3938.1
Coffee	157	345	1.5	9516.1	181	357	1.0	4507.9
Gas + Coffee	162	341	1.5	9516.1	113	247	1.0	3938.1

Table 2.2: Distance between two closest APs in the traces (m).

AP locations: We consider two sets of locations for placing APs: (i) gas stations and (ii) coffee shops. We use Yahoo's Local Search API (version 3) [75] to obtain the longitude and latitude coordinates of 1120 gas stations and 1620 coffee shops in San Francisco Bay Area, as well as 618 gas stations and 738 coffee shops in Seattle. As seen in Table 2.2, the average distance between two closest APs in the traces ranges between $345 - 589$ m and the median distance is $157 - 433$ m. There are quite a few APs whose distance exceeds 3500 m in all the four traces. The communication range between an AP and a vehicle is set to either 100 or 200 meters. We use these values because they approximate the communication ranges we measured from our vehicular testbeds using 802.11b and 802.11g, respectively. To determine the contact period between a vehicle and an AP, we use linear interpolation to obtain the vehicle's mobility trajectory between two adjacent GPS location updates.

Figure 2.3 (c) and (d) illustrate the locations for the gas stations and coffee shops in San Francisco.



(a) CDF of speed of vehicles

(b) CDF of contact time in log-scale

(c) CDF of contact time ranging 0.1s–100s

(d) fraction of total contact time with APs

Figure 2.4: CDF of speed, contact time, and total contact time of 1-day trace

Trace statistics: We compute speed of vehicles based on the GPS coordinates and the time stamps. Figure 2.4(a) plots CDF of speed of vehicles in one day. It shows that 23% – 40% of time the vehicles were parked or moving within 1 miles/hour, 70% of time moving within 11 – 15 miles/hour, and 90% of time moving within 25 – 27 miles/hour. Since most of the cabs are in the

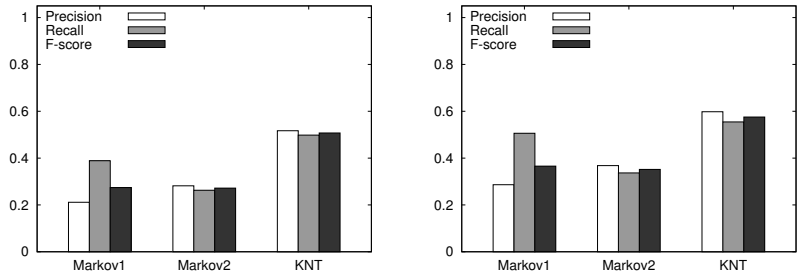
downtown area, they are bounded by the speed limits of the downtown area. Figure 2.4(b) plots CDF of contact duration when APs are placed at either gas stations or coffee shops and the wireless communication range is either 100 or 200 meters. and Figure 2.4(c) further plots CDF of those contact ranging from 0.1 second to 100 seconds. We make the following observations. First, Vehicles in San Francisco have longer contact time with APs at coffee shops than those at gas stations due to more densely populated coffee shops, while the contact time with coffee shops and gas stations is similar in Seattle. Moreover, in all the traces vehicles have short contact time with an AP. In particular, 70% of the contacts between a vehicle and an APs last within 39-51 seconds when wireless range is 100 meters. Figure 2.4(d) further shows percentage of time a vehicle is in contact with an AP for the day. It shows that with communication range of 100 meters, 70% of vehicles spends less than 1.5%-11.5% of time within range of an AP during the day, corresponding to 21–165 minute contact time per day. The short contacts highlight the importance of replicating data in advance.

Baseline algorithms: For baseline comparison, we implement a variant of the mobility prediction algorithm in [52]. The algorithm is based on a second-order Markov mobility model. Each state has two sets of coordinates: the vehicle's location time τ ago, and its current location. In our evaluation, τ is either 1 or 2 or 3 minutes. We deal with irregular GPS update intervals through linear interpolation. To avoid state space explosion, the algorithm discretizes the longitude and latitude coordinates into $0.001^\circ \times 0.001^\circ$ grid

squares. The algorithm uses past mobility traces to learn the probability for a vehicle to transition into any new grid square given its last and current grid squares. Based on the transition probabilities, the algorithm identifies the grid square that the vehicle is most likely to visit next, and uses the center of this grid square as the predicted new location for vehicle after time τ. This procedure is repeated to make predictions further into the future. Based on the predicted locations, the algorithm applies linear interpolation to obtain the entire mobility trajectory and then computes the set of APs the vehicle is predicted to visit during a future interval. As in [65, 52], the algorithm falls back to a first-order Markov model when the second-order Markov model fails to make a prediction. Finally, we also implement the first-order Markov model as another baseline algorithm.

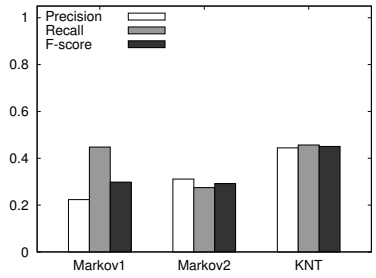
Metrics: We quantify the prediction accuracy using two metrics: (i) *precision*, *i.e.*, the fraction of APs predicted by our algorithms that are indeed visited by the vehicles in a future interval, and (ii) *recall*, *i.e.*, the fraction of APs visited by the vehicles in a future interval that are correctly predicted by our algorithms. To simplify the comparison of different prediction algorithms, we integrate precision and recall into a single metric called *F-score* [73], which is the harmonic mean of precision and recall: $\textit{F-score} = \frac{2}{1/\textit{precision}+1/\textit{recall}}$.

Evaluation results: We consider the following prediction scenario as required by our replication optimization algorithm: *per-interval prediction*, which divides time into fixed intervals and the goal is to predict the set of APs that will be visited by a vehicle in the next interval. The prediction interval is set

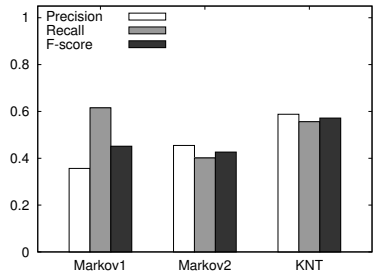


(a) San Francisco, gas, range=100m

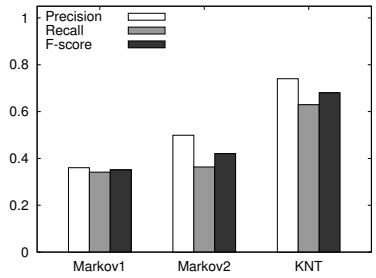
(b) San Francisco, gas, range=200m



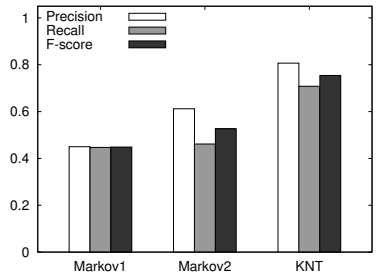
(c) San Francisco, coffee, range=100m



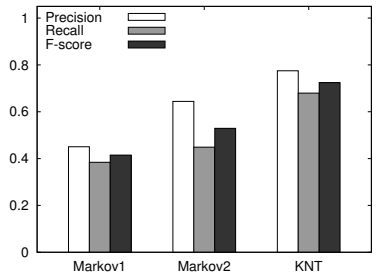
(d) San Francisco, coffee, range=200m



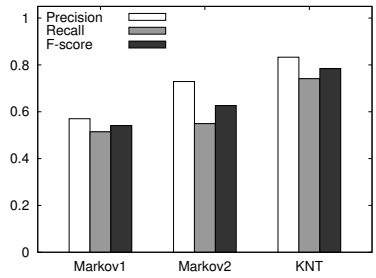
(e) Seattle, gas, range=100m



(f) Seattle, gas, range=200m



(g) Seattle, coffee, range=100m



(h) Seattle, coffee, range=200m

Figure 2.5: Accuracy comparison of different mobility prediction algorithms.

to 3 minutes, which matches the interval for periodic replication optimization. For each prediction algorithm we evaluate, we consider multiple parameter configurations and choose the configuration that yields the best *F-score*. The results from Cabspotting traces use seven days of training data to predict the mobility on the eighth day, and results from Seattle bus traces use 5 days of training data to predict the sixth day as these traces have shorter duration.

Figure 2.5 shows the prediction accuracy when APs are placed at either gas stations or coffee shops and the communication range is either $100m$ or $200m$. For the San Francisco taxi mobility trace (Figure 2.5 (a)–(d)), our algorithm (*KNT*) yields *F-scores* that outperform the first-order Markov model (*Markov1*) and second-order Markov model (*Markov2*) by 25-85%. For the Seattle bus mobility trace (Figure 2.5 (e)–(h)), *KNT* yields *F-scores* that outperform *Markov1* and *Markov2* by 25–94%. In general, the absolute prediction accuracy for all three algorithms is higher for the bus mobility trace, because buses tend to follow fixed routes and are thus more predictable.

Finally, it is worth noting that in contrast to findings in [65, 52], *Markov2* does not significantly outperform *Markov1* in our evaluation. This suggests that with higher speed and less frequent GPS location updates, mobility prediction is more challenging in vehicular networks. As a result, solutions that perform better in less mobile environment do not necessarily perform better in vehicular networks.

Summary: The above results clearly show that our *KNT* mobility prediction algorithm consistently achieves good accuracy in vehicular networks. Later in

Section 2.5, we further show that optimization based on our prediction results yields good performance in practice.

2.5 Trace-Driven Simulation

2.5.1 Simulation Methodology

We develop a trace-driven simulator for evaluation as follows. We first generate the contact traces based on the mobility traces, AP locations, and wireless communication range. When multiple vehicles are within the range of an AP at the same time, we divide the original contacts into non-overlapping contacts, each of which has only one vehicle in contact with an AP. Such contact partitions can be easily realized in practice by letting the AP serve the new vehicle only after it finishes serving the previous one. Similarly, when a vehicle is within the communication range of multiple APs, we also partition the contact into multiple non-overlapping intervals, each of which involves one AP. Another way to partition a contact between multiple vehicles and an AP or multiple APs and a vehicle is to equally divide the contact time among multiple vehicles or multiple APs that are involved in the contact to mimic round-robin scheduling. The performance of these two types of partitions is similar, and we use the first partition in our evaluation.

We then feed the actual contact traces (after the above post processing), predicted contacts, and traffic demands to the simulator. The simulator updates the content at APs and vehicles based on the actual contacts, traffic demands, replication schemes, and wireless and wireline capacity at APs.

We implement network coding for all data transfer to ensure only innovative packets (*i.e.*, whose coding coefficients are linearly independent) are exchanged between APs and vehicles or among APs. We have a content server on the Internet, which has all the content, whereas all APs and vehicles are initialized with no content.

We compare (i) no replication, (ii) wireline replication alone, (iii) vehicular replication alone, (iv) both wireline and vehicular replication, (v) wireline, vehicular, and mesh replication (VCD). In all the schemes, a vehicle downloads content remotely from the Internet whenever the AP has Internet connectivity and the content is not available locally at the AP or mesh network.

To study the impact of traffic demands, we generate traffic demands following either uniform or Zipf-like distribution. In both cases, for every interval, a vehicle randomly selects a specified number of files to request. In the uniform distribution, a file is uniformly drawn from the pool of the files that the vehicle has not requested previously. In Zipf-like distribution, the probability of requesting the ith file is proportional to $\frac{1}{i^{\alpha}}$, where i is the popularity ranking of the file and $i = 1$ indicates the most popular file. We set $\alpha = 0.4$ so that we can generate similar traffic load using both Zipf-like and uniform distributions and the performance difference is solely due to the difference in the distribution.

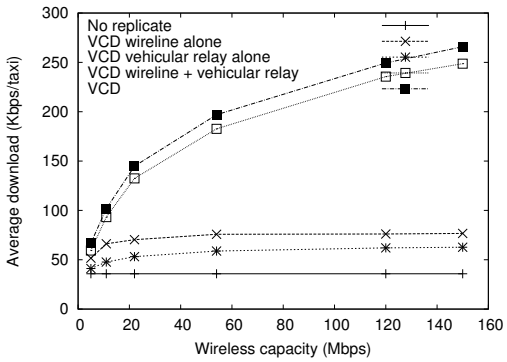
For delay sensitive applications, such as video, their performance depends on the amount of data received before the deadline. Therefore, we use average throughput per vehicle as our performance metric, which denotes the

total demand that is satisfied before the deadline divided by the product of the number of vehicles and the entire trace duration (including the time without contacts with APs). The deadline is set to the end of the interval in which the demand is generated.

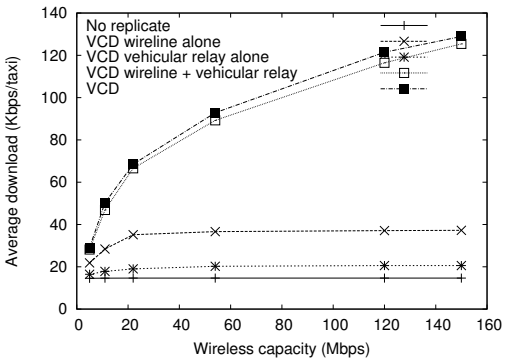
Our evaluation uses 2-hour trace, which exhibits similar contact characteristics as in the 1-day trace, shown in Section 2.4. Other default settings used in our evaluation include: 100-meter communication range between APs and vehicles, 500-meter communication range among APs (well within reach by many mesh routers [4, 48]), Zipf-like traffic demands, placing APs at coffee shops, all APs having 22 Mbps wireless link, half of the APs having Internet links with 2Mbps while the other half have no Internet connection. The content server has a 1 Gbps Internet link and zero wireless capacity to indicate that it is not directly reachable by vehicles. There are 1200 files in total. Each user requests 20 files every 3-minute interval, each file has 2K packets, which contains 1000 bytes. Every file represents either a video clip or one chunk in a larger video file (*e.g.*, We divide a large video file into smaller chunks and generate random linear combinations of packets within each chunk for efficient replication). We further evaluate the effects of changing these parameters.

2.5.2 Simulation Results

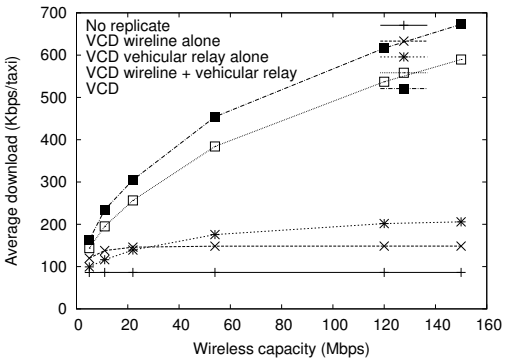
Varying wireless bandwidth: In Figure 2.6, we plot the total downloaded content as we vary wireless bandwidth from 5, 11, 22, 54, 120, and 150 Mbps. We make the following observations. First, in all cases VCD significantly



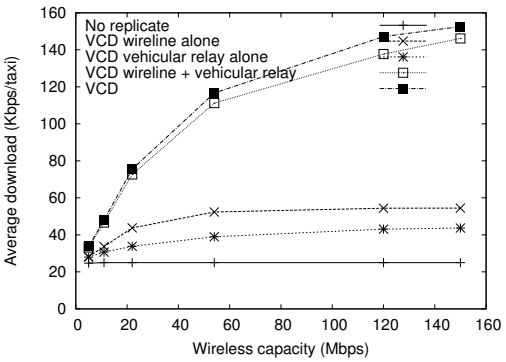
(a) San Francisco, coffee shops, range=100m



(b) San Francisco, **gas station**, range=100m



(c) San Francisco, coffee shops, **range=200m**



(d) **Seattle**, coffee shops, range=100m

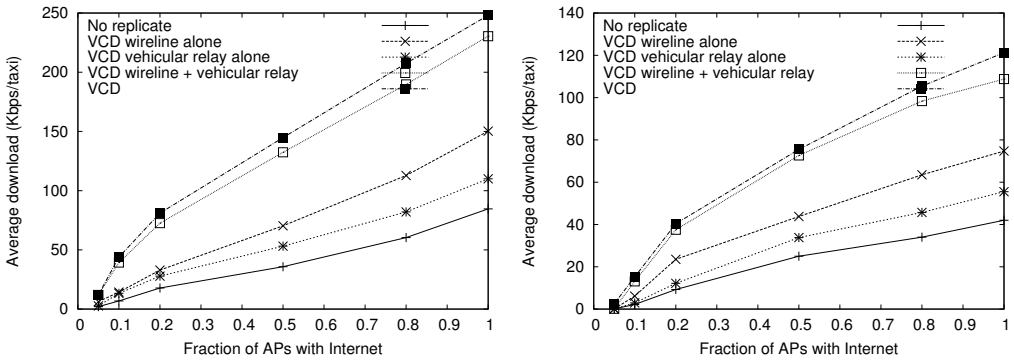
Figure 2.6: Average throughput of 50 cars under varying wireless capacity and Zipf-like traffic demands. The difference from the base configuration is in bold.

out-performs the other schemes and its benefit increases rapidly with wireless capacity. Second, as we would expect, no replication performs the worst. Interestingly, its performance remains the same as we increase wireless capacity. This is because without replication APs often do not have content locally and the wireless download is bottlenecked by slow Internet access capacity. This further demonstrates the need of replication. Third, the performance of

both wireline and vehicular replication alone initially improves with increasing wireless capacity and then tapers off. This is because limited Internet capacity prevents fully taking advantage of large wireless capacity. In comparison, harnessing both wireline and vehicular replication opportunities can effectively utilize the large wireless capacity when available. Adding mesh replication further increases throughput. It improves average throughput by 14-20% under high AP density (Figure 2.6(c)), and by 3-13% in low AP density. Overall, at 22Mbps WiFi capacity, VCD achieves 70 – 300 Kbps average throughput per car depending on the AP density, which can support video streaming applications.

Varying fraction of APs with Internet connectivity: Next we vary the fraction of APs with Internet connectivity. Figure 2.7(a) and (b) plot the average downloaded traffic in San Francisco and Seattle traces, respectively. As we can see, VCD continues to significantly out-perform the other schemes. In addition, the benefits of all types of replication increase with the fraction of APs that have Internet connectivity. The rate of such increase is faster for the replication schemes that involve wireline replication, since they explicitly take advantage of the new wireline capacity to push data.

Varying number of cars: To further evaluate the impact of degree of deployment, we vary the number of cars used from the traces. Figure 2.8 summarizes the performance results. We make the following observations. First, VCD continues to perform the best in all cases. Second, increasing the number of cars initially improves the average throughput because more

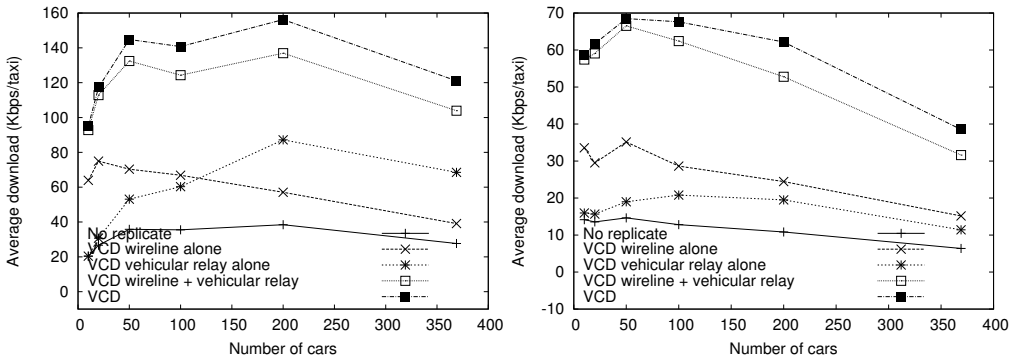


(a) San Francisco (b) Seattle

Figure 2.7: Average throughput under varying fraction of APs with Internet (Zipf-like traffic, APs at coffee shops, range=100, 50 cars).

content are available locally at APs due to previous requests coming from other users. In addition, increasing the number of vehicles also creates more wireless relay opportunities. However, a further increase degrades performance due to increased contention for limited wireline and wireless resources. Third, the benefit of mesh replication increases with the number of vehicles. When we use all the vehicles in the two-hour traces, we find that the mesh replication helps to increase throughput by 17-22%. This is because increasing the number of vehicles increases vehicular relay opportunities and makes it more likely to have content available at nearby mesh nodes.

Varying traffic demands: Figure 2.9 shows the performance for uniformly and Zipf-like distributed traffic demand, respectively. As before, VCD performs the best in all cases. The performance of uniform and Zipf-like distributed traffic receives similar performance. Moreover, decreasing the total number of files tends to improve performance as demands are more concen-



(a) coffee shops (b) gas stations

Figure 2.8: Average throughput under a varying number of cars (San Francisco, Zipf-like traffic, range = 100m).

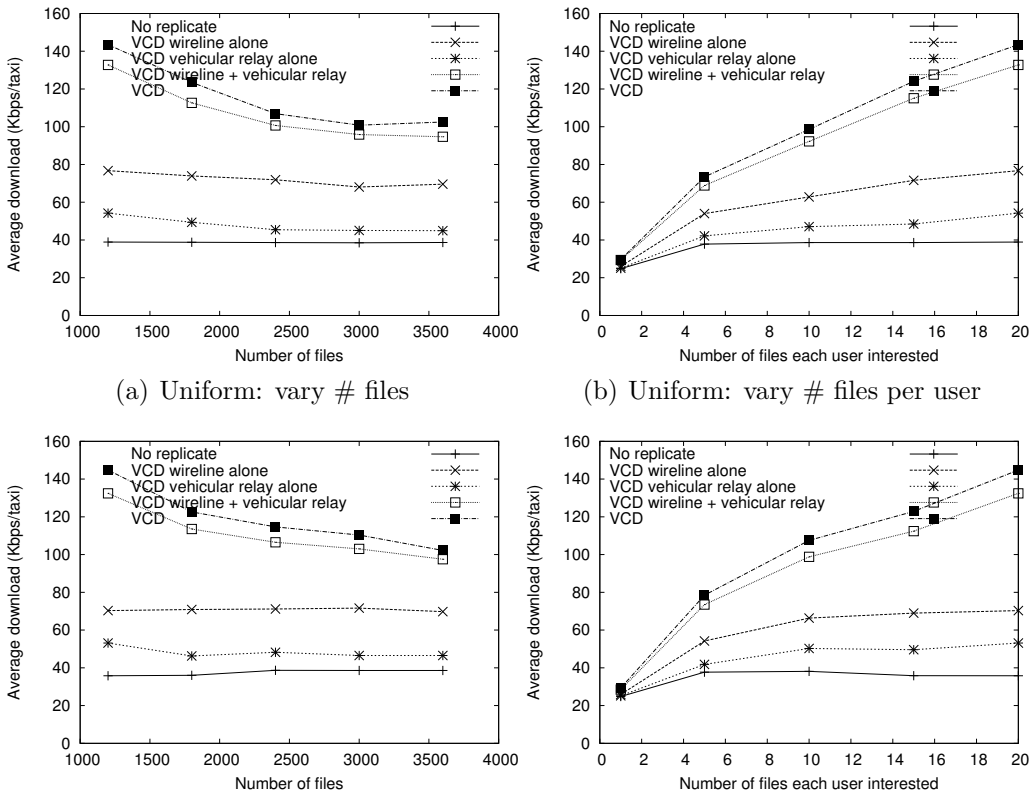
trated and less replication is required to satisfy them. Finally, the replication benefit tends to increase with an increasing number of files requested by each user. This is because when a user is interested in more content, it is more likely to have some locally available content that satisfies the user.

2.6 Trace-Driven Emulation

The goal of our Emulab implementation is twofold: (1) validate simulation results, and (2) evaluate the performance of VCD at scale, which is hard to do in testbed experiments.

2.6.1 Validation

To validate the simulation results, we compare them against those obtained from Emulab under identical settings. We consider the 30 most interactive APs from the trace contacting 100 vehicles. The radio range is 200m.



(a) Uniform: vary # files

(b) Uniform: vary # files per user

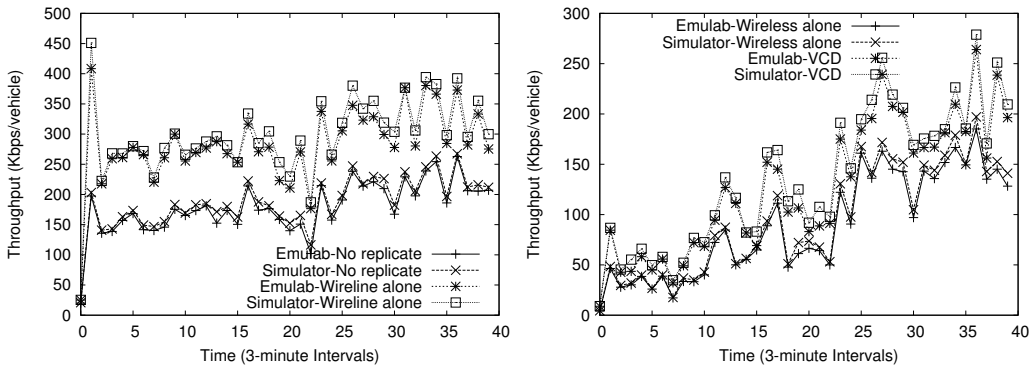
(c) Zipf-like: vary # files

(d) Zipf-like: vary # files per user

Figure 2.9: Average throughput under varying traffic demands (San Francisco, car=50, range=100m, coffee shops).

Given limited machine availability on Emulab, we emulate multiple APs and vehicles on each machine. This limits the link capacity we can select per AP or per vehicle. Hence, our evaluation uses 1Mbps and 6Mbps as the Internet and wireless link capacities, respectively.

Figure 2.10 shows the average throughput for each interval in Emulab and simulator. In Figure 2.10(a), we consider that all APs have Internet connectivity and compare the simulation and emulation performance under no



(a) All APs have Internet (b) 10% APs have Internet

Figure 2.10: Cross validation: comparing performance in Emulab and simulation

replication and wireline replication alone. As it shows, the simulation results closely follow that of Emulab and the discrepancy between them is below 10%. Next we consider that only 10% of the APs have Internet connectivity and compare the performance for vehicular replication alone and VCD in both simulator and Emulab. In this case, since most APs are not connected to the Internet and there is no mesh connectivity, most content is replicated via vehicles. As shown in Figure 2.10 (b), the simulation results match well with Emulab results, within 10% difference for both vehicular replication and VCD.

2.6.2 Micro-benchmarks

The following micro-benchmark results show that our implementation is efficient and light-weight even when operating at scale. We emulate the 120 most interactive APs and 317 vehicles from the trace.

Control message overhead: Table 2.3 shows the per-interval control mes-

Packet type	Avg KB	% of total traffic
Controller to APs	192	0.006
APs to controller	1483	0.048
Content server to AP data	3078200	99.946
Vehicles to APs	49122	1.599
APs to vehicles data	3023100	98.401

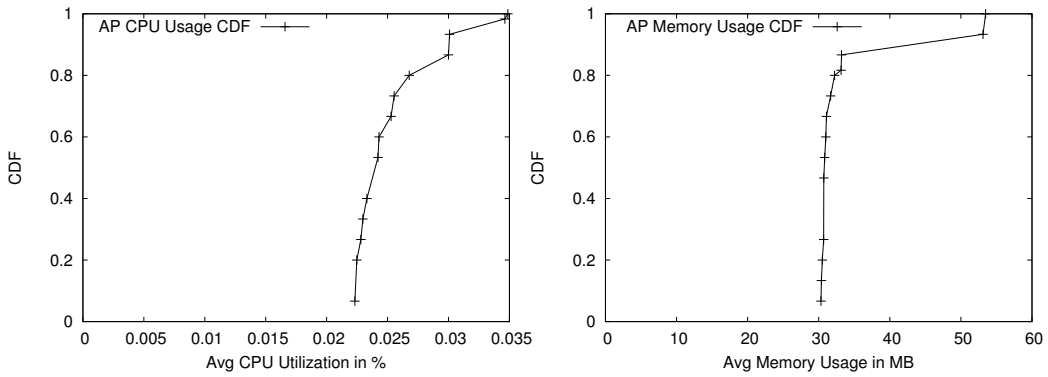
Table 2.3: Average control message overhead per interval.

	Average Latency (ms)
Pre-processing for LP	1307
LP Computation	6512
LP Result Processing	32
Total	7851

Table 2.4: Average controller processing delay per interval

sage overhead imposed by our system. We observe that control messages constitute only 0.054% of the total wireline traffic exchanged amongst APs and between APs and the controller, and constitute only 1.6% of the total wireless traffic between APs and vehicles.

Controller efficiency: We need to ensure that the centralized controller does not become the performance bottleneck. On a 2.133GHz Xeon machine with 3GB RAM, average CPU and memory utilization for the controller is 2% and 38 MB respectively. The average total latency at the controller is 7.8s, which is a small fraction of the 3-minute interval. Table 2.4 further shows the breakdown of the processing latency at the controller. The pre-processing stage involves predicting which APs will be visited and preparing input file for lp_solve. The LP computation, which is performed on Emulab



(a) Average CPU Usage (b) Average Memory Usage

Figure 2.11: CDF of average CPU and memory usage at all APs.

using *lp_solve* [44] due to licensing issues with *cplex* [24], can be further reduced if cplex is used instead.

AP load: Our AP implementation needs to be light-weight and run comfortably on the modest CPU and memory resources available on commercial AP devices. We ran 120 instances of the AP on 2.133GHz Xeon machines with 3GB RAM. Figure 2.11 shows the CDFs of average CPU and memory utilization per AP instance. We find that all APs have roughly the same usage and, on average each AP instance consumes only 0.03% CPU load and 33 MB of memory and hence is light-weight.

2.7 Testbed Experiments

We evaluate our approach using two testbeds. The first testbed consists of 14 APs deployed in office buildings near the road. Due to construction activity, only 9 of these APs could be actively used for experiments. The APs

are Linux desktops equipped with 802.11b radios, which are set to a fixed data rate of 11Mbps. The second testbed consists of 4 APs deployed outdoor equipped with 802.11n radios that use auto-rate. 802.11n radios use 2.4GHz frequency with a 20MHz band. In both testbeds, the APs have 1Mbps wireline access link connecting to the back-end content server. In the 802.11b testbed, 3 out of the 9 APs are connected by a mesh network, whereas all 4 APs in the 802.11n testbed belongs to one mesh network. In both testbeds, mesh communication takes place using additional 802.11b radios. We implement clients on both Windows Mobile Smartphones and Windows XP Laptops. Smartphone clients are used in 802.11b experiments and laptop clients are used in 802.11n experiments. Both clients ran a video streaming application during the car ride. The cars travelled around the testbed at 15 mph (speed limit). We expect that the driving speed does not significantly affect the performance, because increasing speed reduces both on-time (*i.e.*, contact time) and off-time (*i.e.*, the time gap between two consecutive contacts) and these effects should cancel out.

2.7.1 Connection Setup

802.11b: Due to deployment constraints, our AP placement is not ideal: 4 of our APs were placed on the 3rd floor of buildings, limiting their range; and 3 APs were placed in high AP density areas, with 50-70 APs within their range, causing heavy interference. This deployment stress-tests our system. In our experiments during car rides, we were able to associate successfully during

65.2% of all attempts. Most of the failures came from the 3 APs deployed in the high AP density area: association success percentage was only 33.3% for these APs. In fact, even the Windows Mobile WiFi manager utility experienced problems such as very long connection time and adapter freezing near these APs even without any movement. The other access points can successfully associate 85.7% of the time. The association time in our experiments has minimum, median and maximum of 36ms, 844ms, and 14867ms, respectively. 70% of the associations finish within 2 seconds. We retry association up to 7 times and the median retry count is 1.

802.11n: In our 802.11n outdoor testbed, association success rate was 89.58% out of 48 attempts. The minimum, median and maximum association times were 48 ms, 162 ms, and 4086 ms, respectively. 80% of the associations finish within 246 ms and the median retry count was 1. The better results for 802.11n testbed were because we used laptops as clients and placed the APs outdoor (closer to vehicles).

2.7.2 Wireline and Mesh Replication

We implemented a video streaming application that can play H.264 videos encoded at 64Kbps, downloaded from APs. We divide every video into multiple files and use network coding to generate random linear combination of packets within a file. Once enough packets are received for the file, the file is decoded and passed to the video player on the smartphone/laptop to play in proper order using the Windows Mobile media player plugin.

	Download (kB)	Play time (sec)
No replication	29297	3662
Wireline	71930	8991
Wireline + Mesh	79440	9930
Full replication	92493	11562

Table 2.5: Throughput of wireline and mesh replication in the 802.11b testbed

	Download (kB)	Play time (sec)
No replication	16857	2107
Wireline	123175	15387
Wireline + Mesh	130827	16353
Full replication	136479	17060

Table 2.6: Throughput of wireline and mesh replication in the 802.11n testbed

Tables 2.5 and 2.6 compare the performance of our optimized wireline and mesh replication with no replication and full replication at all the APs in 802.11b and 802.11n testbeds, respectively. We report the averages over 3 runs. The full replication assumes every AP has all the files and serves as an upper bound. In both experiments, we follow the planned trajectory, which was fed as input to the controller. In 802.11b testbed, wireline replication alone and wireline plus mesh replication performs 2.45x and 2.7x that of no replication, respectively. In 802.11n testbed, the throughput of wireline and wireline plus mesh replication is 7.3x and 7.8x that of no replication, respectively. This demonstrates the effectiveness of replication. Moreover, the benefit increases with wireless capacity. There is a gap between the performance of VCD and full replication, since the Internet bottleneck prevents complete replication of all the required files.

	No replication		Wireless replication	
	Car 1	Car 2	Car 1	Car 2
AP1	0	0	Upload 780 pkts	Download 780 pkts,20 files
AP2	0	0	Download 1159 pkts, 20 files	Upload 1159 pkts

Table 2.7: Comparison between performance with and without vehicular replication.

2.7.3 Vehicular Replication

To show the benefit of vehicular replication, we use the following setup. Car 1 follows the route $AP1 - AP2$, and Car 2 follows the route $AP2 - AP1$. Car 1 possesses files 1-20 and is interested in files 21-40, while car 2 has files 21-40 and is interested in files 1-20. Both $AP1$ and $AP2$ lack Internet and mesh connectivity. Therefore, without vehicular replication, neither car can get the content it is interested in and the total throughput is 0 under no replication, wireline replication alone, and mesh replication alone.

In comparison, VCD exploits the vehicular replication opportunity. When car 1 meets $AP1$, VCD finds that files 1-20 have highest utility because it predicts car 2 will visit $AP1$ soon and need these files. So $AP1$ instructs the car to upload them first. Similarly, car 2 uploads file 21-40 at $AP2$. When car 1 reaches $AP2$ it can download these files. Similarly, car 2 can download files 1-20 from $AP1$, leading to much higher throughput. Table 2.7 shows that both cars download their interested files in the actual road experiments.

2.8 Related Work

We classify related works into three areas: (i) vehicular networks, (ii) disruption tolerant networks (DTNs), and (iii) mobility and demand prediction.

Vehicular networks: A variety of novel techniques have been proposed to optimize various aspects of communications in vehicular networks. One class of works focuses on techniques for optimizing one-hop communication between a vehicle and nearby APs. For example, CarTel project [18] proposes architectures for vehicular sensor networks, and develops a series of techniques to optimize association, scanning, data transport protocols, and rate selection. ViFi [7] proposes to take advantage of multiple nearby APs to improve communication with passing vehicles. [16] conducts in-depth study of various rate adaptation schemes in vehicular networks and proposes to select data rate based on a combination of RSSI and channel coherence time. [51] uses directional antennas to maximize the transfer opportunity between the vehicle and the AP. These works are complementary to our work, which focuses on end-to-end performance of content distribution. We can potentially leverage these approaches to improve the performance of the last hop. With these enhancements, the gap between Internet and wireless capacity will further increase and make replication even more important. Another class of works consider changes to applications to support vehicular networks. For example, Thedu [6] transforms interactive Web search into one-shot request/response process to reduce access delay. While Thedu still requires connecting with

the remote server, we replicate content to APs to eliminate the Internet bottleneck. The third class of work studies protocol issues. [26] proposes fast connection establishment, scripted handoffs, and prefetching at APs using HTTP range requests. Finally, there are a few works on vehicle-to-vehicle communication. For example, SPAWN [25] uses gossip for file transfer and CarTorrent [40] extends SPAWN and implements it in a testbed. [20] treats vehicular networks as a special type of DTNs and focuses on leveraging vehicle to vehicle (V2V) communication to deliver content. As mentioned earlier, due to the limited V2V contacts [8], we focus on optimizing resource allocation for infrastructure-to-vehicle and vehicle-to-infrastructure communication, which has not been studied earlier.

Disruption tolerant networks: Vehicular networks can also be considered as a special type of disruption tolerant networks (DTNs) and benefit from advances in this area. Different from traditional DTNs, which focuses on communicating with a specific node, we focus on content delivery. Epidemic routing [69] was proposed for DTNs – whenever two nodes meet, they exchange all messages that the other does not have. Recently, utility-based replication was proposed, where nodes replicate data over the best contacts according to some utility (*e.g.*, mobility history [39] or delay [5]). For example, RAPID [5] explicitly tries to optimize system-wide metrics such as average delay while incorporating resource constraints. We leverage both utility based optimization for wireline replication and target wireless replication to achieve efficiency and robustness.

Mobility and demand prediction: There is a large body of literature on mobility prediction, ranging from coarse-grained prediction in cellular networks (*e.g.*, [1, 2, 42, 43, 55]) to more fine-grained prediction in WiFi networks (*e.g.*, [52, 64]). In particular, [65] compares various predictors in literature and suggests that 2nd order Markov with a simple fallback mechanism (when there is no prediction) performs well. [33] builds mobility profiles for users and statistically predicts the next social hub the user will visit. [52] builds the user's customized mobility models on the devices themselves, and uses a second order Markov model to predict the connection opportunity and its quality of the device with an AP. [47] uses the past history to identify opportunities for media sharing in ad hoc DTNs. These works focus on low speed (*e.g.*, personal mobility). Vehicles travel much faster and make mobility prediction more challenging.

Demand prediction has been a well-researched problem in many areas, especially, web content delivery over the Internet [53], and recently over vehicular networks [6]. These works are complementary to our work and we can leverage them to enhance the effectiveness of VCD.

Chapter 3

Incentive-aware Routing in DTNs

We propose an incentive-aware routing protocol for DTNs. When nodes in a DTN are controlled by rational entities, such as people or organizations [5, 11, 19], they can behave selfishly and attempt to maximize their own utility without considering the system-wide criteria. A selfish user may drop others' messages and excessively replicate its own messages to increase its own delivery rate while significantly degrading performance for other users or even causing starvation. Since DTNs have limited connectivity, if any, simply removing selfish nodes results in serious performance penalty. Therefore it is necessary to design incentive-aware routing for DTNs in order to fully take advantage of temporary connections.

While there has been considerable work on studying selfish behavior and designing incentive-aware routing schemes (*e.g.*, [38, 45, 46, 49, 58, 66]), to our knowledge, our work is the first one that studies these issues in DTNs. The lack of contemporaneous path, high variation in network conditions, difficulty to predict mobility patterns, and long feedback delay make the problem very different from the traditional networks like Internet and mobile ad hoc networks. Therefore the existing solutions do not directly apply.

In this chapter, we first study the impact of selfish behavior in DTNs. Using simulation based on both synthetic and real mobility traces, we show that the presence of selfish users can degrade total delivered traffic to less than 20% as what can be delivered under full cooperation.

Motivated by the significant damage caused by selfish users, we propose the use of pairwise tit-for-tat (TFT) as a simple, robust, and practical incentive mechanism for DTNs. Existing TFT mechanisms often face bootstrapping problems or suffer from exploitation. We propose a TFT mechanism that incorporates generosity and contrition to address these issues. We then develop an incentive-aware routing protocol that allows selfish nodes to attempt to maximize their individual utilities while conforming to TFT constraints. We also address the practical challenges involved in implementing the TFT mechanism. We evaluate the effectiveness of our incentive-aware routing scheme using both synthetic and real DTN traces. Our results show that with TFT as a basis of cooperation among selfish nodes, the total delivered traffic increases to 60% or higher as under full cooperation.

Our main contributions can be summarized as follows:

- We study the impact of selfish behavior in DTNs and show that it results in serious performance degradation.
- We develop a practical incentive-aware routing scheme based on the TFT mechanism for selfish users to attempt to optimize their own performance without significant degradation of system-wide performance.

- We demonstrate the effectiveness of our incentive-aware routing scheme using trace-driven simulation.

In this chapter, we first describe related work. We then motivate the need for an incentive mechanism in DTNs in Section 3.2. We study cooperative routing in Section 3.3. Our incentive-aware routing protocol is presented in Section 3.4. Section 3.5 discusses our evaluation methodology. Finally, we present results in Section 3.6.

3.1 Related Work

DTN routing. Routing decisions in DTNs must be made in the absence of end-to-end contemporaneous paths and with limited and possibly stale information about the network. There is a large body of work on routing in DTNs. Most existing schemes are incidental in nature [5]: they do not explicitly optimize a specific performance metric, but opportunistically routes data when temporary connection becomes available. For instance, in epidemic routing [69], whenever two nodes meet, they exchange all messages that the other does not have. Since epidemic routing is essentially flooding, its overhead is high. To reduce the overhead, utility-based replication has been proposed, where nodes replicate data over the best contacts according to some utility (*e.g.*, based on previous mobility [39]). However the effect of how utility relates to the desired performance metrics is unclear. In contrast, RAPID [5] explicitly tries to optimize system-wide metrics such as average delay while incorporating resource constraints, and is shown to be highly effective. Mo-

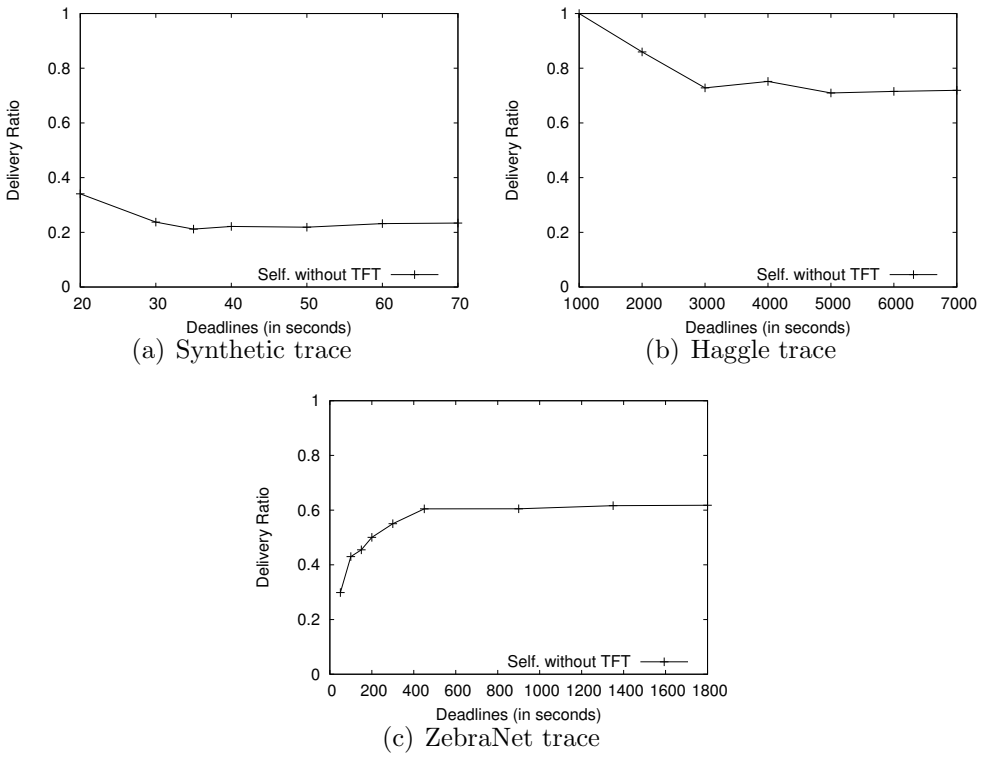
tivated by RAPID, our routing protocol also explicitly optimizes user performance based on the network conditions. However, in contrast to RAPID, which considers only mean link delays, our protocol considers both the mean and variance of link delays and as a result, is robust against high variability in link characteristics. In addition, RAPID only considers global performance objective, whereas our work explores the effects of selfish users in the system.

Incentive mechanisms. Cooperation in the presence of selfish agents has been extensively studied in the Internet, mobile ad-hoc networks, wireless mesh networks, and peer-to-peer applications. Most existing work falls into one of the following three categories. The first category attempts to identify misbehaving nodes and isolate them from the network [45]. These protocols usually assume a set of trusted nodes that can detect and verify misbehavior that results in the selfish node being denied participation in the network. The fear of detection and punishment motivates nodes to cooperate. The second category is based on credits, where nodes earn credits by forwarding packets. These credits can then be used to obtain forwarding service from any node in the system. However, existing credit-based protocols require either secure hardware [13] or trusted centralized banks [77]. In the third category of solutions, nodes reciprocate good or bad behavior on part of the peer in a tit-for-tat fashion [49, 38, 66]. A node autonomously lowers service to a neighbor if it detects that the neighbor is misbehaving, and fully cooperates with the neighbor if no misbehavior is detected. This leads to partial and probably temporary isolation of misbehaving nodes.

We choose tit-for-tat (TFT) as the incentive mechanism for DTN routing. In TFT, every node forwards as much traffic for a neighbor as the neighbor forwards for it. In this way, rather than attempting to detect misbehavior, our approach focuses on detecting good behavior. In our solution, packet acknowledgement acts as the proof of work done by a next-hop. This positive feedback allows a node to engage in balanced exchange with its neighbors—rewarding good behavior with equal reciprocal service and ignoring "misbehavior". In case of the punishment-based approaches, the strong reaction to misbehavior (isolation of the node by peers) is justified only if we have high confidence that the mechanism has very low false positives, so that innocents are not punished, and very low false negatives, so that miscreants cannot get away by flying under the radar. In case of DTNs, there are no reliable ways to detect misbehavior. The existing watchdog mechanism [46] designed for ad hoc networks cannot work in DTNs since it assumes that the sender can listen for the next hop's transmissions to detect if the next hop properly forward the traffic and this assumption fails to hold in DTNs since these two nodes are often disconnected. In addition, due to large variability in mobility patterns and network condition, a node may not be able to deliver a packet within its target deadline in spite of its best intentions. Therefore, the potentially high false positive and false negative in detecting selfish nodes render punishment-based scheme unsuitable for DTNs. In addition, the TFT-based incentive mechanism does not require trusted nodes or special hardware, which fits well with decentralized and low-cost DTNs we envision.

Previous papers (*e.g.*, [38, 49, 66]) have laid the game-theoretic foundation for the use of TFT in wireless networks. The existing TFT mechanisms face bootstrapping problems or suffer from exploitation. We propose a TFT mechanism with generosity and contrition to address these issues. Furthermore, our protocol tolerates significantly large feedback delay in DTNs and supports multi-hop paths (as opposed to single-hop paths in [66]).

TFT has been particularly successful in combating free-riding behavior in P2P file sharing systems. TFT is used to ensure that only agents who actively contribute are allowed to download files from others. Bit-torrent [22] employs a variant of TFT, where k top-performing neighbors in an interval are given equal download rates in the next interval. Analyzing this strategy and improving upon it have been the focus of several recent papers([56, 41]). TFT based file sharing is different from TFT-based routing in DTN in the following ways. First, file sharing is a purely bilateral transaction between two nodes, while routing typically involves interactions among multiple relaying nodes, thus complicating analysis. Second, DTN has large feedback delay and high uncertainty, which makes it critical to address bootstrapping and exploitation. Third, the bilateral nature of file sharing also implies that a neighbor's performance can be evaluated directly, while we must rely on end-to-end acknowledgements to do the same in case of DTN routing. Due to these important differences, we cannot directly apply the TFT mechanism for file sharing to DTN routing.



(a) Synthetic trace

(b) Haggle trace

(c) ZebraNet trace

Figure 3.1: Comparison of fully cooperative DTN against non-cooperative DTN, where the y-axis shows the ratio between delivery rate under a selfish DTN and that under a cooperative DTN.

3.2 Motivation

In this section, we motivate the need for incentive mechanisms by demonstrating that network performance incurs serious degradation without an incentive mechanism. We then identify several important research questions on incentive-aware routing in the DTN context.

The need for incentive mechanism. We compare the performance of fully cooperative DTNs with DTNs consisting of only selfish nodes. Results

for the fully cooperative network are obtained by formulating the DTN routing problem as an Linear Program, which is solved using the CPLEX solver [24]. In the absence of any incentive mechanism to facilitate cooperation, the rational strategy is to free-ride if possible and not relay traffic for anyone else, since there is nothing to be gained by doing so and no notion of penalty in not relaying. If all the nodes in the network follow this strategy, everyone drops relay requests from others implying that packet delivery is only possible when the source directly meets the destination.

We use the fraction of packets delivered within deadline as the performance metric. As shown in Figure 3.1 the delivery rate reduces to only 20% of what is achieved under full cooperation in the synthetic trace, to only 70% in the Haggle trace [60], and to only 60% in the ZebraNet trace [70]. The higher delivery rate in Haggle is due to its higher network connectivity.

These results further confirms our intuition and shows that unless cooperation is somehow incentivized, system operation will be critically impaired. A network of selfish entities needs an incentive mechanism that can act as a basis for such cooperation. As justified in Section 3.1, we choose TFT as the incentive mechanism because it fits well to the unique characteristics of DTNs, such as lack of contemporaneous path, large feedback delay, high variation in network conditions, and unpredictable mobility.

Research questions. In the rest of this chapter, we focus on understanding the following two "prices" in the context of incentive-aware DTN routing.

- **The price of anarchy (PoA)** — Under the given incentive mechanism, what is the performance that can be achieved when all DTN nodes are selfish relative to the optimal performance that can be achieved when all DTN nodes are cooperative?

- **The price of incentive mechanism (PoI)** — Assuming that all users are cooperative, what is the performance penalty that can be achieved under a given incentive mechanism relative to the optimal performance without the incentive mechanism?

PoA and PoI are complementary with each other: PoA quantifies the effectiveness of an incentive mechanism in limiting the damage of selfish nodes, whereas PoI quantifies the performance loss of cooperative nodes due to the presence of the incentive mechanism. In this chapter, we show that with our design, TFT can achieve both low PoA and low PoI for DTN routing.

3.3 Cooperative DTN Routing

We first study the following two cooperative DTN routing schemes. The first scheme optimizes the global objective when everyone is cooperative. This is an interesting baseline since it provides an upper-bound of the performance under TFT constraints. The second scheme optimizes the global objective under TFT constraints. By comparing the performance of these two schemes, we can estimate the PoI of TFT. In the next section, we consider DTN routing under TFT constraints when nodes are selfish.

Routing objective. Throughout this chapter, we consider maximizing total delivered traffic within a given deadline. This is a natural and useful optimization objective. While DTN applications tend to be much more tolerant to delays, it is often useful to be able to impose some application-specific deadline (as opposed to waiting for the delivery for ever). When the deadline is equal to infinity, the objective translates to maximizing the total delivered traffic. Finally, although we only present results for this optimization objective, our algorithms can directly support other optimization objectives, such as minimizing total delay.

3.3.1 Global Optimal

We consider the delivery ratio within a given deadline as the performance metric. The problem of maximizing total delivery ratio within a given deadline over all flows can be solved in the following four steps.

Candidate path generation. First, we generate a set of candidate paths for each given flow by enumerating all possible paths between its source and destination that have at most 3 hops (similar to RAPID [5]). By limiting the length of candidate paths, the number of candidate paths for each flow is bounded by $O(n^2)$, where n is the total number of nodes.

Path performance computation. Next, for each path of a flow, we compute the delivery ratio within a given deadline if the flow is routed through this path. Since the inter-contact time may not follow any well-known distribution, for generality we compute a lower bound of the delivery ratio using

Chebyshev's Inequality [72], which holds for any distribution.

Specifically, we first compute the mean and variance of the waiting time on each link as follows. Given a link between two nodes, we break time into ON periods (in which the two nodes are in contact) and OFF periods (in which the two nodes are not in contact). We assume that if a packet arrives during an ON period, it can be delivered immediately (*i.e.*, the waiting time is 0); if a packet arrives during an OFF period, it can be delivered at the beginning of the next ON period (*i.e.*, the waiting time is equal to the residual time in the current OFF period). For simplicity, we ignore the propagation delay during ON periods because it is typically much smaller than the duration of OFF periods. Assuming that the packet arrival time is uniformly distributed, we can then compute the mean and variance of the waiting time on this link.

We can then approximate the delivery ratio (within a given deadline) for an end-to-end path as follows. For any given path, let random variable X denote the total waiting time. Let μ and σ^2 denote the mean and variance of X. μ and σ^2 can be computed by summing up the mean and variance of the waiting times on different links on this path (under the assumption that these waiting times are independent). Let D denote the desired deadline. Then according to Chebyshev's Inequality, the delivery ratio within the deadline D

can be bounded as follows:

$$\begin{aligned} Pr(X \leq D) =& 1 - Pr(X \geq D) \\ =& 1 - Pr(X - \mu \geq D - \mu) \\ \geq& 1 - \left(\frac{\sigma}{D - \mu}\right)^2 \end{aligned} \tag{3.1}$$

Route optimization. Then we maximize the total delivery ratio within the deadline for all flows by formulating the problem as the linear program (LP) shown in Figure 3.2. Here $X_{f,i}$ is the traffic allocation of flow f on path i; $P_{f,i}$ is the lower bound of the delivery ratio when traffic of flow f is routed through path i given by Inequality (3.1); Cap_i denotes the smallest capacity of all links on path i. Constraint C1 specifies the capacity constraint, *i.e.*, the total amount of traffic routed through path i should not exceed the capacity of path i. Constraint C2 mandates that the total traffic assignment for flow f does not exceed the demand of flow f by a factor of $RepFactor$, where the replication factor $RepFactor$ is a control parameter that can be tuned to improve the total delivery ratio at the cost of more replication traffic. In our evaluation, we keep $RepFactor$ constant at 3.

Online optimization. Finally, to deal with fluctuations in traffic demands and network connectivity, we break time into segments and perform route optimization at the beginning of each segment based on predicted traffic demands and path performance. Specifically, we use the exponentially weighted moving average (EWMA) of past values to predict the traffic demands and the mean and variance of path waiting time for the current segment. We then use these

$Input : Flows, Demand(f), P_{f,i}, Cap_i$
$Output : X_{f,i}$
max: $\sum_{f \in Flows} \sum_{i} X_{f,i} P_{f,i}$
Subject to:

[C1] $\sum_{f \in Flows} X_{f,i} \leq Cap_i \quad \forall i$

[C2] $\sum_{i} X_{f,i} \leq RepFactor * Demand(f) \quad \forall f$

Figure 3.2: LP formulation to maximize delivered traffic within a given deadline.

predicted values in the above LP formulation to optimize the routes for the current segment. The routes remain unchanged until the beginning of the next segment.

3.3.2 Global Optimal with TFT Constraints

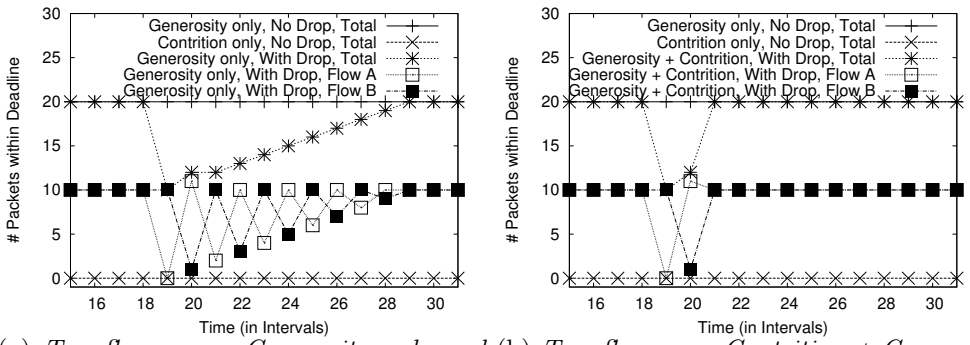
Next we incorporate the TFT mechanism when maximizing total delivered traffic. This can be achieved by adding the following TFT constraints into the LP shown in Figure 3.2. The TFT constraints simply state that the total amount of traffic through link $A \rightarrow B$ is equal to the total amount of traffic in the opposite direction (*i.e.*, through link $B \rightarrow A$).

$$\sum_{f} \sum_{i:\, AB \in Path_i} X_{f,i} = \sum_{f} \sum_{j:\, BA \in Path_j} X_{f,j} \quad \forall nodes\ A, B \tag{3.2}$$

TFT constraints with generosity: The basic TFT constraints (as de-

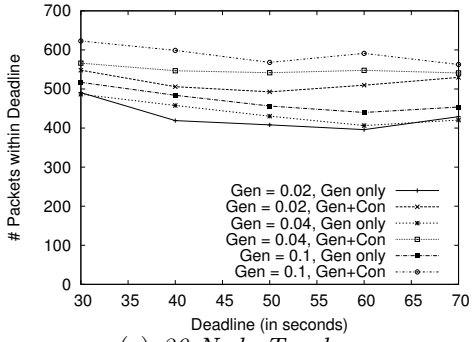
scribed above) have problems with bootstrapping. When two nodes meet for the first time, since no packets have ever been successfully relayed by the other node, the basic TFT prevent any relaying. To address this issue, generous TFT enables initial cooperation of up to ϵ, which allows a node to send ϵ number of packets more than it has earned the right to send by relaying in the previous interval. Generous TFT is also useful for handling asymmetric traffic demands by absorbing traffic imbalance up to ϵ amount. The ϵ value is important. A larger value loosens TFT constraints and yields better performance and faster bootstrapping when everyone is cooperative. On the other hand, it also means that a selfish node is less cooperative since this generosity can be exploited by the node. We model exploitation as follows. Every selfish node checks if it has performed enough work in the previous interval to be able to satisfy its predicted demand for the upcoming interval without requiring any generosity from the neighbor. If not, it has incentive to provide generosity in order to get increased service in the next interval. Otherwise (*i.e.*, if its predicted service rate is no less than its predicted demand), it has no incentive to provide any generosity to that neighbor. In addition, it assumes that the neighbor will provide it generosity. As a result, it does ϵ less work and can get away with it if the other node indeed does provide ϵ generosity. In Section 3.6, we empirically study the impact of generosity.

TFT constraints with generosity and contrition: Generosity alone is still insufficient. While it can absorb transient asymmetry in delivery of up to ϵ, any imbalance exceeding ϵ could lead to lengthy retaliation between two



(a) *Two-flow case. Generosity only and Contrition only* (ϵ = 1)

(b) *Two-flow case. Contrition + Generosity* (ϵ = 1)



(c) *20 Node Topology*

Figure 3.3: Effect of Generosity, Contrition and Generosity + Contrition. (ϵ = 1)

neighbors. This is illustrated in the following example.

Consider a toy topology of four nodes with two flows $A \rightarrow B \rightarrow C$ and $B \rightarrow A \rightarrow D$, each with a demand of 10 packets/interval. When network connectivity is stable, generosity allows relays A and B to gradually increase their cooperation until their demands are satisfied after which both will attempt to exploit the other. However, generous TFT can cause protracted vendetta between neighbors if one of them delivers less packets than expected for the other, possibly due to variation in mobility. This is demonstrated in

Figure 3.3 (a), where outage on link $B \rightarrow C$ in interval 19 causes zero packets to be delivered for A's flow in that interval. Since this drop in delivery exceeded ϵ, A retaliates in the next interval by delivering correspondingly less. In interval 21, generous TFT constraints require B to retaliate back, which sets off a long-lasting vendetta.

Contrition solves this problem by refraining from reacting to a valid retaliation to its own mistake. With contrite TFT, B realizes that A's reaction in interval 20 was due to its own actions in interval 19, and so does not lower traffic in interval 21. This way cooperation is restored from interval 21. In addition, since contrition does not always provide leeway like generous TFT, it cannot be exploited. However, contrition cannot work by itself either in case of DTN routing. Contrition only provides a way to return to stability after perturbation, but provides no way to reach that stability, *i.e.*, as Figure 3.3 (a) shows, it does not provide any way for cooperation to bootstrap like generosity does.

Hence, we propose a novel variant of TFT by combining generosity and contrition. The generosity component enables bootstrapping and absorbs transient asymmetries, while contrition prevents mistakes from causing endless retaliation. Figure 3.3 (b) shows that by interval 21, contrition and generosity working together allow total delivery to recover from the outage in interval 19.

In Figure 3.3 (c), we show that the above argument remains valid even for a larger topology of twenty nodes. We compare the number of packets delivered for different values of generosity and different packet deadlines. Packet

delivery increases with increasing generosity as TFT conditions are loosened to accommodate asymmetric demand. More importantly, TFT with both generosity and contrition prevents unnecessary retaliation and outperforms TFT with generosity alone by up to 30% in some cases.

3.4 Selfish DTN Routing

The previous section describes methods for optimizing routes when all nodes in a DTN are cooperative. In this section, we present a practical distributed protocol that allows selfish users to optimize their individual performance while conforming to TFT constraints.

Our routing protocol consists of the following three components: (i) every node periodically exchanges link state, (ii) each source computes the forwarding paths based on link state and uses source routing to send its traffic, (iii) upon receiving data, each destination sends ACK via flooding and the source uses it to update its TFT constraints for the next interval.

Link state dissemination: Every node keeps track of the mean and variance of the waiting time on links between the node itself and other nodes. It also computes the link capacity using the duration of the meeting and the bandwidth available during that time. At the end of every interval, every node floods these three link metrics so that all nodes have the information about all links in the network. This is similar to many link state protocols, such as OSPF. We assume that link state is disseminated faithfully—we focus on making the data-plane incentive compatible and leave securing the control

plane to future work.

Route computation: We use source routing to send traffic. This gives a source complete routing control so that it can directly optimize its own performance metric. Moreover, if different senders are interested in optimizing different performance metrics (*e.g.*, some want to minimize delay and others want to maximize delivery rate), this can be easily supported using source routing. In order to prevent the source route from being tampered in transit, it is digitally signed by the sender (*e.g.*, using Hierarchical Identity Based Cryptography (HIBC) [62], which is shown to be practical for DTNs [63]).

Based on the link state, a source maximizes its total delivered traffic as follow. For each data packet, a source generates $RepFactor$ number of packets and specifies complete source route for each generated packet. The routing strategy is computed at the beginning of every interval. Given average delay, variance, and link capacity of each link (which is disseminated throughout the network), a source first computes the delivery ratio within a given deadline for each path using Equation 3.1. The end-to-end ACKs (as described below) indicate how many packets are successfully delivered on each path for each flow during the previous interval. Then it can compute the total background traffic along each path in the previous interval, and estimate the background traffic volume information in the new interval using ACK packets of data delivery. Next it updates its routing strategy in the new interval by moving traffic from the worst path to the best path in terms of delivery ratio. The move continues until either link capacity constraint or TFT constraint is violated.

Then it starts to move traffic to the second best path and so on until all the paths with better delivery rate have reached their raw link capacity or TFT constraints. Figure 3.4 outlines the algorithm.

```
1  src: node ID of flow f's source
2  X_{f,i}^{prev}: amount of traffic from flow f is allocated on path i in the previous interval
3  X_{f,i}^{curr}: amount of traffic from flow f is allocated on path i in the current interval
4  B_{f,i}^{prev}: amount of background traffic on path i in the previous interval
5  compute the lower-bound of the delivery ratio along each path using Equation 3.1
6  sort paths in the order of increasing delivery ratio
7  worst = 1; best = totalPaths
8  while P(worst) < P(best)
9      // compute maximum amount of flow f's traffic that can be sent on path i
10     TFTCap(best) = infinity;
11     for each relay link k on path best
12         TFTCap(best) = min(TFTCap(best), forwarded(k));
13     end
14     cap = min(TFTCap(best), Cap(best)) - B_{f,i}^{prev};
15     // we can move up to moveTotal traffic from the worst to best path
16     moveTotal = min(X_{f,worst}^{prev}, cap);
17     X_{f,worst}^{curr} -= moveTotal;
18     X_{f,best}^{curr} += moveTotal;
19     if X_{f,best}^{curr} == cap
20        best = best - 1;
21     end
22     if X_{f,worst}^{curr} == 0
23        worst = worst + 1;
24     end
25 end
```

Figure 3.4: Route computation at selfish nodes.

Note that $TFTCap$ in Figure 3.4 is based on the total traffic others have forwarded in the previous interval, denoted as $forwarded(k, src)$. However, as all TFT-based schemes, the actual TFT constraints should be based on the total traffic sent during the current interval. Moreover the background traffic along each path may also change over different time intervals. Therefore the route derived above may not satisfy the actual TFT constraints. To address the issue, we apply the following dropping strategy. Let $T_{i,j}$ and $T_{j,i}$ denote the total traffic node i relays for j and the total traffic node j relays for i in

the current interval. If $T_{i,j} > T_{j,i} + \epsilon$, node i ensures TFT by dropping traffic from j that exceeds $T_{j,i} + \epsilon$ packets.

ACK dissemination: Upon receiving a packet, the destination floods an ACK so that we can use it to derive the TFT constraints for the next interval. In our protocol, acknowledgements serve the following three purposes.

- First, packet acknowledgements generated by the destination act as proofs of successful relay by intermediate nodes. Once the packet reaches its destination, the destination node extracts the source route and the accompanying signature and attaches them to the acknowledgement packet. The ACK packet is then flooded through the network. The size of ACK is much smaller than the size of data traffic, so the ACK overhead should be small. To further improve efficiency, we can combine multiple ACK packets into a single ACK during the flooding.

- Second, acknowledgements can provide useful feedback required by TFT. Specifically, every node receiving the ACK first verifies the integrity of the attached source route and then checks if its identifier is present in the relay list. If it is, then the node increments its local TFT counters to indicate that the next node in the list successfully relayed a packet for it. Credit is only given to relay nodes on the forwarding path.

- Third, flooded acknowledgements disseminate key information about the network operation to every node. Each node uses information provided in the ACK to compute how many packets were sent between every source

and destination pair and along which paths. Since the ACK contains the entire source route, a node can also calculate the number of packets traversing every link in both directions.

3.5 Evaluation Methodology

We implement the routing protocol described in Section 3.4 in dtnsim from [37]. For comparison purpose, we also evaluate the routing computed by solving LP as described in Section 3.3 using CPLEX [24]. We compare different routing schemes by varying mobility traces and traffic demands. In all evaluation, we set link capacity to 10 packets/second.

Mobility traces: We use synthetic mobility traces to gain insights to the routing schemes under controlled scenarios, and use real traces to assess their performance under realistic scenarios. We generate synthetic traces as follows. We have 20 nodes and randomly create 114 links among them. We then generate the ON/OFF time for each link, where the ON time is kept constant at 1, and the OFF time follows a Gaussian distribution whose mean and variance are 10 and 0.5, respectively. In addition to the synthetic traces, we also use the Haggle trace [60] which involves 41 iMotes carried by IEEE INFOCOM attendees and the ZebraNet trace consisting of the movement of 20 male zebras in a 6km-by-6km field each carrying a radio with a range of 500m, generated using the same methodology as [70].

Traffic demands: To study the impact of traffic demands on the routing

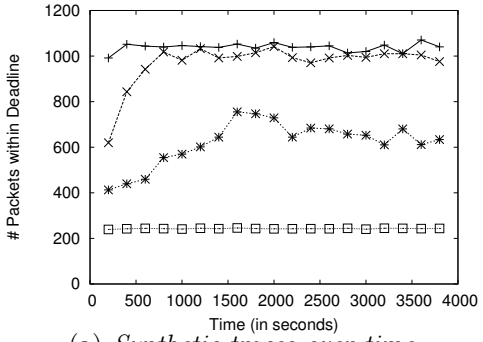
performance, we generate traffic demands as follows. We first randomly generate 5 flows originating from each node. We then set the total traffic demands from all nodes to be either all equal or following a Zipfian distribution. In Zipfian distribution, the top i-th demand from a node is proportional to $1/i$. We use Zipfian distribution, because a number of studies show that realistic user demands often exhibit Zipf-like distributions [10, 21]. Finally, we partition the total traffic demand at a node to all its flows either equally or using Gravity model [76]. In Gravity model, the total traffic from A to B is proportional to the total outgoing traffic from A and the total incoming traffic to B. In this way, we have four spatial distributions: (i) *equal/equal*, (ii) *equal/gravity*, (iii) *Zipf/equal*, and (iv) *Zipf/gravity*.

3.6 Evaluation Results

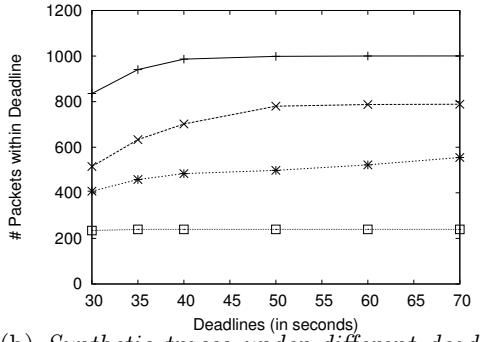
We evaluate the performance of our routing scheme by varying the mobility, traffic demands, and deadlines.

Impact of different traces: First, we focus on one time segment in each trace and assume that we know the mean and variance of ON/OFF time of links between every pair of nodes during the time segment. In the later evaluation, we will consider the effects of prediction errors.

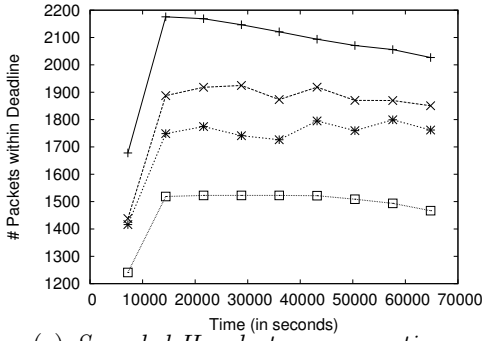
Figure 3.5(a), (c), and (e) plot the total number of delivered packets within the deadline over time for the synthetic, Haggle, and ZebraNet traces, respectively. We set the deadline to be 70 seconds for the synthetic trace, 7000 seconds for Haggle trace, and 1350 seconds for ZebraNet trace. As we



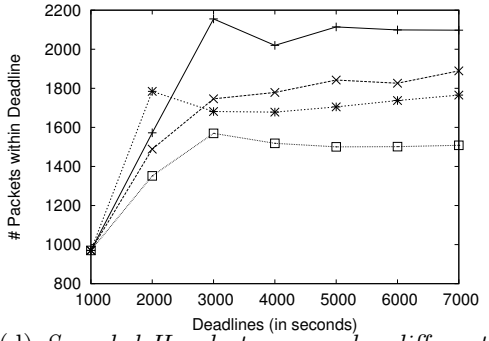
(a) *Synthetic traces over time*



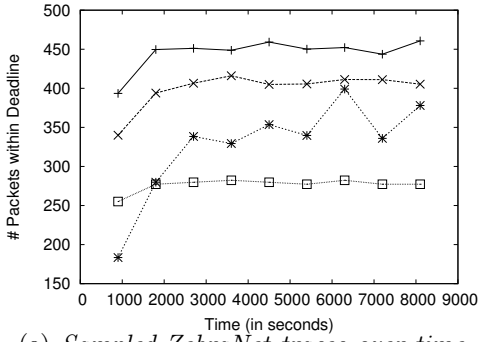
(b) *Synthetic traces under different deadlines*



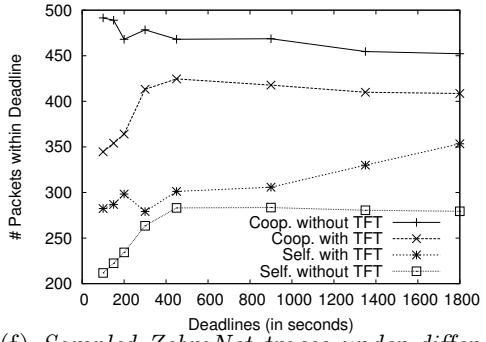
(c) *Sampled Haggle traces over time*



(d) *Sampled Haggle traces under different deadlines*



(e) *Sampled ZebraNet traces over time*



(f) *Sampled ZebraNet traces under different deadlines*

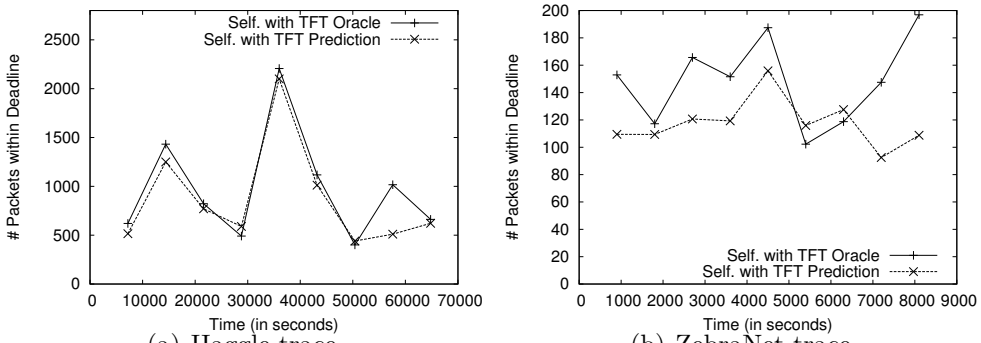
Figure 3.5: Impact of different traces.

would expect, cooperation without TFT achieves the highest delivery rate, since there is neither PoA nor PoI. Cooperation with TFT performs the second best and its difference from that without TFT is within 7-20% for all traces, which suggests that the TFT incentive mechanism imposes low cost in the presence of cooperation and therefore has low PoI: PoI of 20% for synthetic, 10.5% for Haggle, and 7% for ZebraNet trace. The performance of selfish users with TFT also achieves low PoA compared to cooperative counterpart: PoA of 25%, 6%, and 15% respectively. Finally, we observe selfish users without TFT performs the worst, because without an incentive mechanism delivery is only possible when source and destination directly meet. We note that in Figure 3.5(e) Selfish with TFT performs worse than selfish without TFT in the first interval. This is because the latter delivers only packets destined for the node in contact, while the former also spends bandwidth forwarding multi-hop packets over a contact. This causes fewer packets to be actually delivered to their destinations in the first interval but helps set stage for TFT cooperation in later intervals leading to higher delivery.

We demonstrate the effect of deadlines in Figure 3.5(b), (d), and (f), which plot the average number of packets delivered within the deadline. The results are the average over all intervals after the bootstrap phase is over, so Figure 3.5(b), (d), and (f) are the average of 15, 7, and 8 runs, respectively. As we would expect, packet delivery increases with the deadlines since packets have more time to reach the destination. Moreover, TFT performs well for a wide range of deadline values. The only exception is using 1000 second

deadline in Haggle trace, where all routing schemes perform similarly. This occurs because the deadline is too small to allow multi-hop delivery and the only possible delivery even under full cooperation is through direct contact.

From the above comparison of traces, we found that the performance gap among different cooperation schemes varies according to the nature of the traces. The rank of the schemes, however, remains the same regardless of the difference in the mobility of the traces, indicating that the TFT incentive mechanism is beneficial for various networks.



(a) Haggle trace (b) ZebraNet trace

Figure 3.6: Comparison of inter-meeting prediction versus oracle under temporal variation in mobility. EWMA parameters: $\alpha = 0.8$, EWMA interval = trace interval

Impact of temporal variation in mobility: We now study the impact of temporal variation in mobility by using the entire raw Haggle and ZebraNet traces. Since the Haggle trace captures the mobility of participants at a computer science conference, it consists of periods of high interactivity (presumably indicating lunch or break times), interspersed with low activity periods

(presumably indicating conference sessions). In case of the ZebraNet trace, animal movements dictate how and when communication can take place. In the presence of such variation in mobility, we need to estimate the mean, variance of delay and the bandwidth for every link in order to compute the source routes.

We compare two estimation schemes. First, we propose a prediction scheme that uses EWMA values of link characteristics as estimate for the next interval. Second, we use values provided by an oracle that has knowledge of the true values for the next interval. The oracle represents the best possible estimation.

Figure 3.6 compares the two estimation methods. For the Haggle trace (Figure 3.6(a)), we observe that the prediction scheme performs within 10% of the oracle which is reasonably accurate given the high variation in the mobility for the three days of conference. While the ZebraNet trace results (Figure 3.6(b)) do not show any discernible diurnal pattern, it also has high variability in the node's movements over time. Moreover, the EWMA interval size and the total volume of the packets are order of magnitude less than those of Haggle traces, making EWMA predictor to follow the oracle's line with 21% error.

Comparing the oracle and prediction results, we can observe that with EWMA-based prediction, the results are reasonably close to perfect prediction. However, we note that predicting link properties is an interesting and open problem for DTNs.

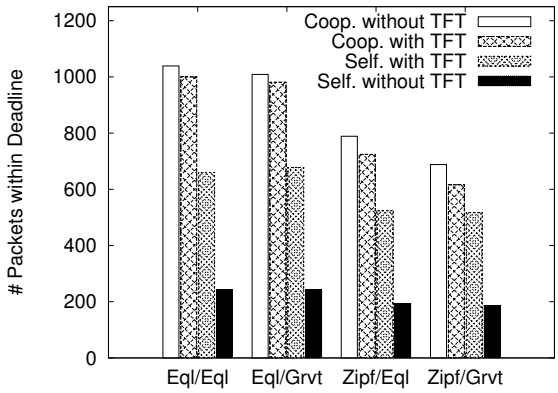
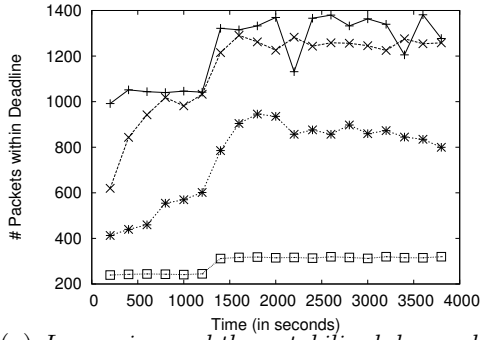


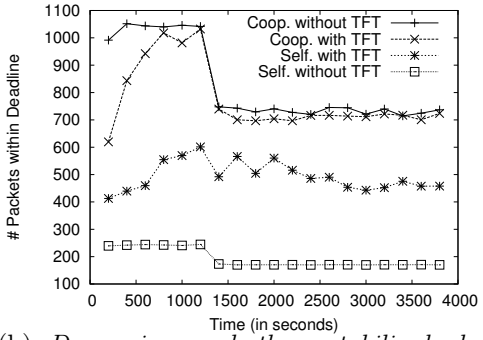
Figure 3.7: Comparison of algorithms under spatial demand difference.

Impact of spatial variation in traffic demands: Next we study the impact of spatial variation in traffic demands using the synthetic trace. Figure 3.7 shows the total delivery rates for different spatial distributions: equal/equal, equal/gravity, Zipf/equal, and Zipf/gravity, which are described in Section 3.5. Here the number of packets delivered within deadline represents an average of 15 runs. We observe that the relative performance across the different routing schemes remains the same. Moreover, the PoI under a cooperative DTN is around 5%. In addition, without the incentive mechanism, the delivery rate in a selfish DTN is only around 200 packets/second. In comparison, with the incentive mechanism, the delivery rate increases to 500 packets/second for Zipf-distributed demands, and 600 packets/second for equal demands. These results demonstrate the effectiveness of our incentive-aware routing scheme.

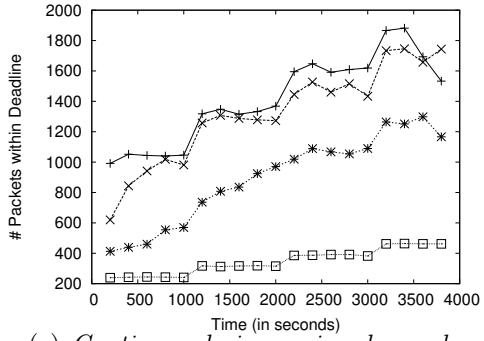
Impact of temporal variation in traffic demands: Finally we compare routing schemes by varying the traffic demands over time using the synthetic



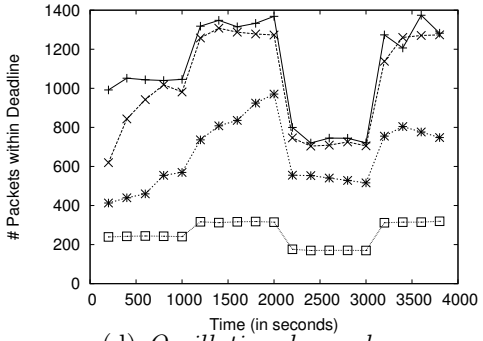
(a) *Increasing and then stabilized demands*



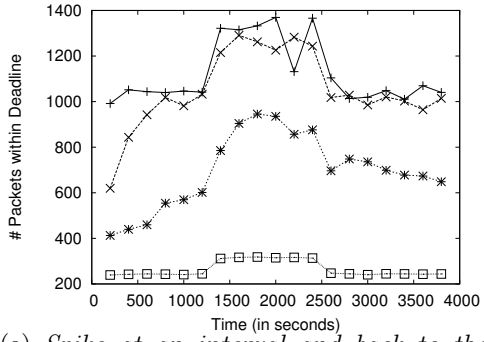
(b) *Decreasing and then stabilized demands*



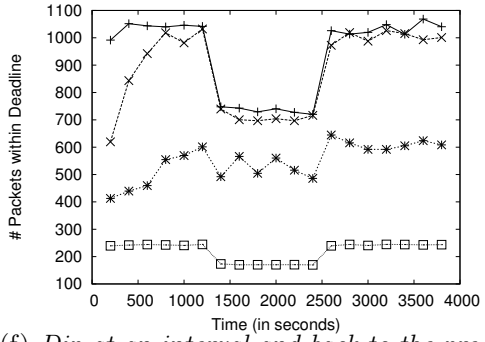
(c) *Continuously increasing demands*



(d) *Oscillating demands*



(e) *Spike at an interval and back to the previous demand level*



(f) *Dip at an interval and back to the previous demand level*

Figure 3.8: Comparison of algorithms under different temporal variations.

trace. The goal is to understand the responsiveness of different routing schemes to demand changes, *i.e.*, whether the algorithm converges after disruption in demand traffic, and if so, how long it takes for the delivery ratio to stabilize.

We consider the following six traffic variation scenarios: (a) increasing and then stabilized demands, (b) decreasing and then stabilized demands, (c) continuously increasing demands, (d) oscillating demands, (e) spike at an interval and going back to the previous demand level, and (f) dip at an interval and going back to the previous demand level. In all the cases, every flow has the same demand (*i.e.*, their spatial distribution follows equal/equal).

As shown in Figure 3.8, the relative performance of different routing schemes are consistent across different temporal demand variations. The price of incentive mechanism is small, and the incentive mechanism significantly improves delivery rate in a selfish DTN. In addition, the delivery rate adapts quickly with the change in traffic demands in all cases.

Chapter 4

Conclusion

In this dissertation, we address performance and incentive issues in disruption-tolerant networks.

First, we present the VCD system that provides high-bandwidth content access in vehicular DTNs by utilizing opportunistic connections to WiFi access points along the road. VCD predicts which APs a vehicle will encounter in the future and proactively pushes content to these APs by leveraging both wireline and wireless connectivity. Using trace-driven simulation and Emulab-based emulation, we show that VCD is capable of downloading 3-6X as much content as no replication and 2-4X as much content as wireline or vehicular replication alone. The gap further increases as the ratio between wireless and wireline capacity increases. We further develop a full-fledged prototype of VCD using two testbeds: a 14-AP 802.11b testbed and a 4-AP 802.11n testbed with smartphone and laptop clients. Our experience suggests that VCD is an effective approach for vehicular content distribution.

Second, we study the impact of selfish behavior in DTNs and show that it results in serious degradation in routing performance. We then propose the use of tit-for-tat (TFT) as a simple, robust and practical incentive

mechanism for DTNs. Making TFT practical for DTNs is challenging due to the lack of contemporaneous end-to-end paths, high variation in network conditions, difficult to predict mobility patterns, and long feedback delay in DTNs. Existing TFT mechanisms often face bootstrapping problems or suffer from exploitation in such environment. We therefore propose a TFT mechanism that incorporates generosity and contrition to address these issues. We then develop a an incentive-aware routing protocol that allows selfish users to adaptively optimize their individual performance subject to TFT constraints. We also address the practical challenges involved in implementing the TFT mechanism. Using both synthetic and real DTN traces, we show that our incentive-aware routing protocol is effective in fostering cooperation among selfish nodes and can significantly improve the routing performance.

Future work:

- **Cost-conscious combination of 3G and Wifi.** Smartphones today are equipped with both 3G and WiFi interfaces. While WiFi provides intermittent high-bandwidth connectivity, 3G allows always-on but lower-bandwidth and high-cost path to the Internet. This complementarity between the two technologies can be exploited to design a cost-conscious hybrid replication strategy.
- **General currency system as a incentive mechanism.** While TFT provides a basis for rational nodes to engage in routing, the performance under TFT is limited since TFT credits acquired with a neighbor can only

be used to send packets through that neighbor. A general currency systems, where credit earned can be used to buy service from any other neighbor, can potentially provide better performance. However, implementing a currency system in a DTN in the absence of an always-available central bank or use of trusted hardware is an interesting open problem.

Bibliography

[1] I. F. Akyildiz and W. Wang. The predictive user mobility profile framework for wireless multimedia networks. *IEEE/ACM Transactions on Networking*, 12(6), 2004.

[2] A. R. Aljadhai and T. Znati. Predictive mobility support for QoS provisioning in mobile wireless environments. *IEEE Journal on Selected Areas in Communications*, 19(10):1915–1930, 2001.

[3] AT&T DSL. http://www.att.com/gen/general?pid=6431.

[4] AWE Mesh Router. http://www.nomadio.net/AWE_overview.pdf.

[5] A. Balasubramanian, B. Levine, and A. Venkataramani. DTN routing as a resource allocation problem. In *Proceedings of ACM SIGCOMM*, pages 373–384, 2007.

[6] A. Balasubramanian, B. Levine, and A. Venkataramani. Enhancing interactive web applications in hybrid networks. In *Proceedings of ACM MobiCom*, Sept. 2008.

[7] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. Levine, and J. Zahorjan. Interactive WiFi connectivity for moving vehicles. In *Proceedings of ACM SIGCOMM*, 2008.

[8] N. Banerjee, M. Corner, D. Towsley, and B. Levine. Relays, base stations, and meshes: Enhancing mobile networks with infrastructure. In *Proceedings of ACM MobiCom*, Sept. 2008.

[9] BMW car2car communication development. http://www.motorauthority.com.

[10] L. Breslau, P. Cao, L. Fan, G. Philips, and S. Shenker. Web caching and Zipf-like distributions: evidence and implications. In *Proceedings of IEEE INFOCOM*, Mar. 1999.

[11] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. MaxProp: routing for vehicle-based disruption-tolerant networks. In *Proceedings of IEEE INFOCOM*, pages 1–11, Apr. 2006.

[12] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss. Delay-tolerant networking: an approach to interplanetary internet. *IEEE Communications Magazine*, 41(6):128–136, 2003.

[13] L. Buttyán and J.-P. Hubaux. Enforcing service availability in mobile ad-hoc wans. In *Proceedings of ACM MobiHoc*, pages 87–96, Piscataway, NJ, USA, 2000. IEEE Press.

[14] V. Bychkovsky, B. Hull, A. Miu, H. Balakrishnan, and S. Madden. A measurement study of vehicular Internet access using in situ Wi-Fi networks. In *Proceedings of ACM MobiCom*, pages 50–61, 2006.

[15] Cabspotting. http://www.cabspotting.com.

[16] J. Camp and E. Knightly. Modulation rate adaptation in urban and vehicular environments: Cross-layer implementation and experimental evaluation. In *Proceedings of ACM MobiCom*, Sept. 2008.

[17] Car2Car communication consortium. http://www.car-to-car.org.

[18] Cartel. http://cartel.csail.mit.edu/doku.php.

[19] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of Human Mobility on the Design of Opportunitic Forwarding Algorithms. In *Proceedings of IEEE INFOCOM*, Apr. 2006.

[20] B. B. Chen and M. C. Chan. MobTorrent: a framework for mobile internet access from vehicles. In *Proceedings of IEEE INFOCOM*, 2009.

[21] M. Chesire, A. Wolman, G. M. Voelker, and H. M. Levy. Measurement and analysis of a streaming-media workload. In *Proceedings of the 3rd USENIX Sympsoium on Internet Technologies and Systems*, Mar. 2001.

[22] B. Cohen. Incentives Build Robustness in BitTorrent. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, 2003.

[23] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of ACM MobiCom*, Sept. 2003.

[24] CPLEX. http://www.ilog.com/products/cplex/.

[25] S. Das, A. Nandan, and G. Pau. Spawn: a swarming protocol for vehicular ad-hoc wireless networks. In *Proceedings of IEEE VANET*, 2004.

[26] P. Deshpande, A. Kashyap, C. Sung, and S. Das. Predictive methods for improved vehicular wifi access. In *Proceedings of ACM MobiSys*, 2009.

[27] DARPA Disruption Tolerant Networking Program. http://www.darpa.mil/ato/solicit/dtn.

[28] BBN Technologies Awarded $9M to Add Disruption Tolerance to Military Networks. http://www.bbn.com/news_and_events/press_releases/2008_press_releases/pr_dtn_contract_award.

[29] Emulab. http://www.emulab.net.

[30] J. Eriksson, H. Balakrishnan, and S. Madden. Cabernet: Vehicular content delivery using WiFi. In *Proceedings of ACM MobiCom*, Sept. 2008.

[31] K. Fall, G. Iannaccone, J. Kannan, F. Silveira, and N. Taft. A disruption-tolerant architecture for secure and efficient disaster response communications. In *ISCRAM (Information Systems for Crisis Response and Management)*, 2010.

[32] Gartner research. http://www.gartner.com/it/page.jsp?id=1210613.

[33] J. Ghosh, M. J. Beal, H. Q. Ngo, and C. Qiao. On profiling mobility and predicting locations of wireless users. In *Proceedings of REALMAN*, 2006.

[34] M. Grossglauser and D. N. C. Tse. Mobility increases the capacity of ad hoc wireless networks. In *Proceedings of IEEE INFOCOM*, Apr. 2001.

[35] D. Hadaller, S. Keshav, T. Brecht, and S. Agarwal. Vehicular opportunistic communication under the microscope. In *Proceedings of ACM MobiSys*, 2007.

[36] T. Ho, M. Medard, J. Shi, M. Eros, and D. R. Karger. On randomized network coding, Oct. 06 2003.

[37] S. Jain, K. Fall, and R. Patra. Routing in a Delay Tolerant Network. In *Proceedings of ACM SIGCOMM*, pages 145–158, New York, NY, USA, 2004. ACM.

[38] J. J. Jaramillo and R. Srikant. DARWIN: distributed and adaptive reputation mechanism for wireless ad-hoc networks. In *Proceedings of ACM MobiCom*, pages 87–98, New York, NY, USA, 2007. ACM.

[39] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In *Proceedings of ACM ASPLOS*, Oct. 2002.

[40] K. C. Lee, S.-H. Lee, R. Cheung, U. Lee, and M. Gerla. First Experience with CarTorrent in a Real Vehicular Ad Hoc Network Testbed. In *Proceedings of MOVE*, 2007.

[41] D. Levin, K. LaCurts, N. Spring, and B. Bhattacharjee. Bittorrent is an auction: analyzing and improving bittorrent's incentives. In *Proceedings of ACM SIGCOMM*, 2008.

[42] G. Liu and G. Maguire, Jr. A class of mobile motion prediction algorithms for wireless mobile computing and communication. *Mobile Networking Applications*, 1(2), 1996.

[43] T. Liu, P. Bahl, S. Member, and I. Chlamtac. Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks, Mar. 1998.

[44] Lp_solve: Linear programming code. http://www.cs.sunysb.edu/∼algorith/implement/lpsolve/.

[45] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Sustaining cooperation in multi-hop wireless networks. In *Proceedings of USENIX NSDI*, pages 231–244, Berkeley, CA, USA, 2005. USENIX Association.

[46] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *Proceedings of ACM MobiCom*, Aug. 2000.

[47] L. McNamara, C. Mascolo, and L. Capra. Media sharing based on colocation prediction in urban transport. In *Proceedings of ACM MobiCom*, pages 58–69, 2008.

[48] Meraki MR58. http://meraki.com/products_services/access_points/MR58/.

[49] F. Milan, J. J. Jaramillo, and R. Srikant. Achieving cooperation in multihop wireless networks of selfish nodes. In *Proceedings of workshop on Game theory for communications and networks (GameNets)*, page 3, New York, NY, USA, 2006. ACM.

[50] Mobile Broadband Review 2010. http://mobile-broadband-services-review.toptenreviews.com/.

[51] V. Navda, A. P. Subramanian, K. Dhanasekaran, A. Timm-Giel, and S. Das. Mobisteer: using steerable beam directional antenna for vehicular network access. In *Proceedings of ACM MobiSys*, pages 192–205, 2007.

[52] A. J. Nicholson and B. D. Noble. Breadcrumbs: Forecasting mobile connectivity. In *Proceedings of ACM MobiCom*, Sept. 2008.

[53] V. N. Padmanabhan and J. C. Mogul. Using predictive prefetching to improve world wide web latency. *ACM SIGCOMM Computing Communication Review*, 26(3), 1996.

[54] J. Partan, J. Kurose, and B. N. Levine. A Survey of Practical Issues in Underwater Networks. *Special Issue of ACM Mobile Computing Communications Review*, 11(4):23–33, October 2007.

[55] P. N. Pathirana, A. V. Savkin, and S. Jha. Mobility modelling and trajectory prediction for cellular networks with mobile base stations. In *ACM MobiHoc*, 2003.

[56] M. Piatek, T. Isdal, T. E. Anderson, A. Krishnamurthy, and A. Venkataramani. Do Incentives Build Robustness in BitTorrent? In *Proceedings of USENIX NSDI*, 2007.

[57] Laptop & smartphone users prefer Wi-Fi to 3G; willing to pay for citywide Wi-Fi. http://www.teleclick.ca/2009/02/laptop.html.

[58] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker. On Selfish Routing in Internet-Like Environments. In *Proceedings of ACM SIGCOMM*, Aug. 2003.

[59] Qualcomm Innovation - WiFi offload. http://www.qualcomm.com/documents/3gwifi-seamless-offload.

[60] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau. CRAWDAD data set cambridge/haggle (v. 2006-09-15). Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/haggle, Sept. 2006.

[61] Seattle Bus Traces. http://crawdad.cs.dartmouth.edu/meta.php?name=rice/ad_hoc_city.

[62] A. Seth and S. Keshav. Practical security for disconnected nodes. In *Proceedings of NPSEC*, 2005.

[63] A. Seth, D. Kroeker, M. Zaharia, S. Guo, and S. Keshav. Low-cost Communication for Rural Internet Kiosks Using Mechanical Backhaul. In *Proceedings of ACM MobiCom*, Sept. 2006.

[64] L. Song, U. Deshpande, U. C. Kozat, D. Kotz, and R. Jain. Predictability of WLAN mobility and its effects on bandwidth provisioning. In *Proceedings of IEEE INFOCOM*, 2006.

[65] L. Song, D. Kotz, R. Jain, and X. He. Evaluating location predictors with extensive Wi-Fi mobility data. In *Proceedings of IEEE INFOCOM*, Mar. 2004.

[66] V. Srinivasan, P. Nuggehalli, C. Chiasserini, and R. Rao. Cooperation in wireless ad hoc networks. In *Proceedings of IEEE INFOCOM*, volume 2, pages 808–817 vol.2, 2003.

[67] Memory cards. http://en.wikipedia.org/wiki/Comparison_of_memory_cards.

[68] Toyota and Honda vehicle-to-vehicle communication systems. http://www.motorauthority.com/toyota.html.

[69] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University, 2000.

[70] Y. Wang, P. Zhang, T. Liu, C. Sadler, and M. Martonosi. CRAWDAD data set princeton/zebranet (v. 2007-02-14). Downloaded from http://crawdad.cs.dartmouth.edu/princeton/zebranet, Feb. 2007.

[71] Wifi-taxi. http://www.wifi-taxi.com/cgv_en.htm.

[72] Wikipedia. Chebyshev's inequality. http://en.wikipedia.org/wiki/Chebyshev's_inequality.

[73] Wikipedia. F1 score. http://en.wikipedia.org/wiki/F1_score.

[74] Worldwide pricelist for iPhone 3G Plans. http://www.unwiredview.com/2008/07/16/worldwide-pricelist-for-iphone-3g-plans/.

[75] Yahoo! Local Search Web Services. http://developer.yahoo.com/search/local/V3/localSearch.html.

[76] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale IP traffic matrices from link loads. In *Proceedings of ACM SIGMETRICS*, 2003.

[77] S. Zhong, J. Chen, and Y. Yang. Sprite: a simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *Proceedings of IEEE INFOCOM*, volume 3, pages 1987–1997, Mar.-Apr. 2003.

Vita

Upendra Shevade was born on October 1, 1981 in Kolhapur, India, second child of Shailaja and Bhalchandra Shevade. He attended Navi Mumbai High School, and later Swami Vivekanand Junior College. He received Bachelor of Engineering in Information Technology from the University of Mumbai in 2003. In Fall 2003, he joined the University of Texas at Austin as a graduate student. He received Master of Arts in Computer Sciences in December 2005 and Doctor of Philosophy in Computer Science in August 2010. He is looking forward to where life will take him now.

Permanent address: 'Gajanan Prasad', Plot B55,
Sector 23, Nerul
Navi Mumbai, India 400706

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.