DATA DISSEMINATION IN VEHICULAR AD HOC NETWORKS

A Dissertation in
Computer Science and Engineering
by
Jing Zhao

© 2008 Jing Zhao

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2008
The dissertation of Jing Zhao was reviewed and approved* by the following:

Guohong Cao  
Associate Professor of Computer Science and Engineering  
Dissertation Advisor, Chair of Committee

Thomas F. La Porta  
Distinguished Professor of Computer Science and Engineering

Chita R. Das  
Distinguished Professor of Computer Science and Engineering

Sencun Zhu  
Assistant Professor of Computer Science and Engineering

Xiaolong Zhang  
Assistant Professor of Information Science and Technology

Raj Acharya  
Professor of Computer Science and Engineering  
Department Head of Computer Science and Engineering

*Signatures are on file in the Graduate School.
Abstract

Vehicular ad hoc networks (VANET) have been envisioned to be useful in road safety and many commercial applications. Although VANET is a kind of mobile ad hoc network, many unique characteristics of VANET bring out new research challenges. For example, due to fast vehicle movement, the link topology changes rapidly. As a result, many well studied structures for efficient data dissemination such as tree, cluster and grid, are extremely hard to be set up and maintained. Also, the network density is highly dynamic. The traffic density is low in rural areas and during night, which may result in frequent disconnections and network partitions. Thus, data dissemination techniques should address these unique characteristics of VANET.

The specific goal of this dissertation is to provide cohesive solutions for data dissemination in VANETs. Different algorithms and protocols are applied based on where the vehicles are and how far they are from the roadside units. First, we design and evaluate schemes for vehicles far away from the roadside units. To deal with the frequent disconnection problem in sparsely connected VANET, we adopt the idea of carry and forward. Different from existing works, we make use of the predictable mobility in VANET, which is limited by the traffic pattern and the road layout. Second, we design and evaluate pushed-based solutions for vehicles near the roadside units. To reduce the bandwidth consumption, data poured from the source are buffered and rebroadcasted at the intersections. Analytical models are developed to provide guidelines to maximize the dissemination capacity. Third, we propose a relay-based solution for vehicles approaching the one-hop range of a roadside unit. With relay from vehicles close to the roadside unit, vehicles faraway can still access the roadside unit with high rate rate. Finally, we study the benefit of using the vehicular infrastructure mesh network to disseminate data. We design and implement a cooperative cache based scheme which allows mesh stations to cooperatively cache data for vehicles within the mesh network coverage.
# Table of Contents

List of Figures viii  
List of Tables x  
Acknowledgments xi  

Chapter 1  
**Introduction**  
1.1 Motivation ................................................. 2  
1.2 Challenges for Data Dissemination in VANETs ............... 3  
1.3 Focus of This Dissertation  
1.3.1 Data Access over Large Scale Sparsely Connected VANETs . 5  
1.3.2 Data Pouring and Buffering in Densely Populated VANETs 6  
1.3.3 Extending The Coverage of Roadside Unit ............... 7  
1.3.4 Cooperative Caching in Vehicular Infrastructure Mesh Networks ............................................. 7  
1.4 Organization .................................................. 9  

Chapter 2  
**Related Work**  
2.1 MAC Layer .................................................. 10  
2.2 Network Layer ................................................ 11  
2.3 Application Layer .......................................... 12  
2.4 Internetworking with Roadside Units  

Chapter 3  
**Data Dissemination over Large Scale Sparsely Connected VANETs** 15  
3.1 Introduction ............................................... 15
4.4.1 The Simulation Model ........................................... 57
4.4.2 Simulation Results .................................................. 61
  4.4.2.1 The relation between delivery ratio and dissemination capacity ........................................... 61
  4.4.2.2 Revisiting the relations between delivery ratio, broadcast cycle time and dissemination capacity 62
  4.4.2.3 The effects of bit error rate ........................................... 64
  4.4.2.4 Dissemination distance ........................................... 64
  4.4.2.5 Dissemination delay ........................................... 65
  4.4.2.6 Network traffic overhead ........................................... 66
4.5 Conclusions ............................................................... 68

Chapter 5
Extending The Coverage of Roadside Units 69
5.1 Introduction ............................................................... 69
5.2 Motivations ............................................................... 71
5.3 The Vehicle-to-Vehicle Relay (V2VR) Scheme ........................................... 74
  5.3.1 Assumptions ............................................................... 74
  5.3.2 Scheme Overview ............................................................... 74
  5.3.3 Step 1: A Stochastic Model for Finding A Reliable Proxy Group .................................................. 76
  5.3.4 Step 2: Forward Relay .................................................. 81
    5.3.4.1 Establishing the forward relay connection .................................................. 81
    5.3.4.2 Connecting to a RSU .................................................. 82
  5.3.5 Step 3: Backward Relay .................................................. 83
5.4 Implementation ............................................................... 84
  5.4.1 Background ............................................................... 84
  5.4.2 Information Flow ............................................................... 85
  5.4.3 Experimental Results .................................................. 88
5.5 Performance Evaluations .................................................. 89
  5.5.1 Simulation Setup .................................................. 89
  5.5.2 Simulation Results .................................................. 90
    5.5.2.1 RSU Connection Duration .................................................. 90
    5.5.2.2 Amount of Data Delivered .................................................. 91
    5.5.2.3 Data Transmission Throughput .................................................. 92
5.6 Conclusions ............................................................... 93
# List of Figures

3.1 Find a path to the coffee shop ........................................... 18
3.2 The transition modes in VADD ........................................... 18
3.3 An example of the VADD Delay Model ................................. 20
3.4 One road graph ............................................................. 21
3.5 Add a boundary ............................................................. 22
3.6 Select the next vehicle to forward the packet ......................... 24
3.7 A scenario of routing loop ............................................... 25
3.8 A snapshot of the simulation setup area ............................... 33
3.9 Data delivery ratio as a function of the data sending rate .......... 34
3.10 Percent of data packets dropped due to routing loops or MAC layer packet collisions (150 nodes) .......................... 36
3.11 Data delivery delay as a function of the data sending rate ......... 37
3.12 The number of packets generated .................................... 39
3.13 Impact of data packet size .............................................. 40

4.1 Directional Broadcast ..................................................... 45
4.2 Analytical model for $t_q$ ................................................ 53
4.3 Analyze the relation between the broadcast cycle time and the data delivery ratio ........................................ 56
4.4 A snapshot of the simulation setup area ................................ 58
4.5 A snapshot of the simulation setup area ................................ 58
4.6 Comparison of the data delivery ratio (with the broadcast cycle of 10s) ......................................................... 60
4.7 Relations among cycle time, data set size and delivery ratio ...... 63
4.8 Impact of the transmission bit error rate ............................... 64
4.9 Delivery ratio at different distances from the data center ........ 65
4.10 Delivery delay at different distance from the data center .......... 66
4.11 Comparison of the network traffic .................................... 67
4.12 Detailed comparison of the network traffic .......................... 67

5.1 Experiment setup .......................................................... 72
5.2 Beacon delivery ratio and UDP throughput at different distance from the RSU. ................................................. 73
5.3 Access RSU through a relay. ..................................... 75
5.4 A Markov model for three-lane roadway. ...................... 79
5.5 Transition matrix for three-lane Markov model. ............ 79
5.6 Select a relay proxy. .............................................. 80
5.7 The traditional 802.11 and V2VR RSU access timeline. .... 84
5.8 Hello packet format. The packet contains the minimum amount of information required for a trailing vehicle to use another vehicle to forward its traffic. .............................................. 85
5.9 The decision making process and information flow of the implementation. We define being associated as the traditional definition, plus having a valid IP address and an acceptable SNR. If the vehicle can verify that it is associated, it will use the RSU. Otherwise, it will use the vehicle that it chose to act as its forwarder for passing traffic to the RSU. .............................................. 86
5.10 The sequence of events for a client to use a proxy. The proxy broadcasts its hello packets and goes through a typical association process with a RSU (1 – 15). Once association is complete, the proxy’s next broadcast includes DNS information (16). After receiving status change and DNS address, the client uses the proxy to relay traffic (17 – 22). .............................................. 87
5.11 Complementary CDF of the connection duration. The mean duration for the no-relay scheme and the relay scheme is 6.06 and 9.68 respectively. 90
5.12 Average TCP traffic transmitted per vehicle .................. 91
5.13 Vehicle TCP data transmission throughput at different distances. . 92

6.1 A wireless mesh infrastructure network. ....................... 97
6.2 Layered design .................................................. 100
6.3 In the asymmetric approach, the data reply only goes up to the cache layer at the intermediate nodes that need to cache the data. 102
6.4 The greedy algorithm for determine the cache placement. ...... 109
6.5 CCSL design .................................................. 110
6.6 Topology of the testbed ........................................ 110
6.7 Cache hit ratio of the three approaches at different data sizes. ... 112
6.8 The data access delay comparison in 802.11 based mesh networks . 116
6.9 A close view of query latency for different cooperative cache schemes in single-interface-single-channel 2M 802.11 based mesh networks .... 116
6.10 Comparison of the data access delay in MIMC mesh networks ...... 118
6.11 Comparison of the data traffic generated in 5M MIMC mesh networks 118
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Simulation Setup</td>
<td>33</td>
</tr>
<tr>
<td>4.1</td>
<td>Simulation Setup</td>
<td>60</td>
</tr>
<tr>
<td>5.1</td>
<td>Notations</td>
<td>77</td>
</tr>
<tr>
<td>5.2</td>
<td>Simulation Setup</td>
<td>89</td>
</tr>
<tr>
<td>6.1</td>
<td>Data access delay (in milliseconds)</td>
<td>111</td>
</tr>
<tr>
<td>6.2</td>
<td>Packet processing time at the cache layer</td>
<td>112</td>
</tr>
<tr>
<td>6.3</td>
<td>Wireless interface setup</td>
<td>113</td>
</tr>
<tr>
<td>6.4</td>
<td>Simulation Parameters</td>
<td>115</td>
</tr>
<tr>
<td>6.5</td>
<td>Simulated data access delay using the five-node topology</td>
<td>115</td>
</tr>
</tbody>
</table>
Acknowledgments

I would like to express my sincere gratitude to my advisor Dr. Guohong Cao for his inspiration, patience, encouragement and invaluable guidance that lead me to a deeper understanding of knowledge work. I am grateful for the numerous meetings that he had with me to discuss our research problems and guide me through them with his profound knowledge in many diverse areas. He is always there when I need advice or help. Without his tremendous support and constant guidance, I could not have completed this dissertation.

I am grateful to Dr. Thomas F. La Porta and Dr. Chita R. Das for the inspiration and insightful comments on my thesis, which ensured me to work on the right track towards finishing my dissertation. I want to thank Dr. Sencun Zhu for the inspiring conversations on my research and for providing advices on issues of my school life and career. I also want to thank Dr. Xiaolong Zhang for providing valuable comments and giving me enlightening guidance as I started my dissertation.

My gratitude also goes to my co-author Yang Zhang for inspiring conversations and insightful suggestions to help me to work through our research problems. I would like to thank my co-authors Ping Zhang and Todd Arnold for their hard work that led to the development of our system. Special thanks to Hui Song, Changlei Liu and Min Shao, for being wonderful group mates and friends, for the encouragement on my work and for making my years of study at Penn State an enjoyable and memorable journey.

Last but not least, I am deeply indebted to my parents and brother for their unconditional love and unending support during my school years. I also wish to express my deepest gratitude to my wife for her constant faith in me and sacrifices that she made during my candidature.

This work was supported in part by the National Science Foundation under grant CNS-0721479.
Dedication

To Ying, my wife.
Chapter 1

Introduction

It has only been less than a century since the first automobile came to the world. Vast innovations in safety, comfort, and convenience have already made today’s vehicle a very different machine than it was in the past century. Now a new technology characterized by proliferation of low-cost wireless connectivity and distributed peer-to-peer cooperative systems, is changing the way in which next generation vehicle will evolve. This new technology is the vehicular ad hoc network or VANET. It brings people with new features and applications that have never been possible.

A VANET is an ephemeral, rapidly changing wireless network formed among vehicles and roadside units which are able to communicate with one another. To participate in a VANET, vehicles must be equipped with wireless transceivers and computerized control modules that allow them to act as network nodes. Stationary network nodes may be used in the form of roadside units. These roadside units may help with a wide variety of services for vehicular networks such as hosting location-relevant data, or serving as a gateway to the Internet. To realize this vision, FCC has allocated 75 MHz of spectrum for dedicated short range communications (vehicle-vehicle or vehicle-roadside) [1]. IEEE is currently working on the IEEE 802.11p Wireless Access in Vehicular Environments (WAVE) standard [2, 3]. The standard will provide a multi-channel DSRC solution with high performance for multiple application types to be used in future VANETs. As more and more vehicles are equipped with communication capabilities that allow for intervehicle communication, metropolitan-wide VANETs are expected to be available in the near future.
1.1 Motivation

VANETs have been envisioned to be useful in numerous applications, serving the interests of consumers, businesses, governments, and emergency services [4, 5, 6, 7, 8]. It is generally believed that three areas will receive great benefits from the use of VANETs. The first area centers on improving the safety of motor vehicles. VANET can be used to propagate emergency warning of numerous conditions that can cause a collision, including vehicles ahead braking quickly, vehicles swerving to avoid road obstructions, erratic lane changing and etc.. In case an accident does occur, communication made to other vehicles prior to collision may allow the accident to be reconstructed more easily. The second area receiving direct benefit is relevant to transportation traffic control. The immediate benefit from VANET is to improve the efficiency of traffic system. Information about traffic jam can be acquired in real-time so that drivers heading towards the congested area can receive it with sufficient time to choose alternate routes. Toll roads can be automatically paid without the installation of additional hardware to a vehicle. Traffic signals equipped with communication equipment can be used to more accurately control intersection traffic. The third beneficial area is to enhance the convenience of drivers and add elements of pleasure for the travelers. Through the use of VANETs and widely deployed wireless WiFi's or infostations[9], location relevant local shopping and service information can be provided in real-time. For example, useful local advertisements and announcements can be delivered to travelers, such as sale information at a department store; the available parking lot at a parking place; the room availability and price at a hotel; the menu at a restaurant. From radio to television and video players, the possibilities for in-vehicle entertainment have been constantly evolving. Many new applications can be made possible with VANETs, such as Internet access for passengers, information exchange and multimedia content sharing with passengers in other vehicles.

All the aforementioned applications rely on the metropolitan wide VANETs which involve moving vehicles and fixed roadside units. The roadside units can come in many forms and provide many different functions, such as information broadcast station to announce roadside services, data collection hub to collect real-time traffic information, WiFi access point and metro mesh node to provide
Internet access. Data are disseminated in the network by employing all possible connectivities including vehicle to vehicle communication, vehicle to roadside communication and roadside to roadside communication. The wireless communication range of a network node is limited from several hundred meters to a few thousand meters. So data are often required to hop through multiple nodes across various geographical distances before successfully delivered to the intended recipients.

VANET, as one form of ad hoc network, shares many properties of a general mobile ad hoc network. Many data dissemination protocols have been developed for mobile ad hoc networks (MANETs) [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21], However, they cannot be directly used, because many unique characteristics of VANET bring out new research challenges. In the next section, we present these challenges for data dissemination in VANETs.

1.2 Challenges for Data Dissemination in VANETs

The biggest challenge in VANET is that the network mobility and topology at different parts of metropolitan area are highly diverse: some vehicles may move at less than 5 miles per hour in a traffic jam, while others move at over 75 miles per hour on highways; The traffic density is low in suburban areas and during night, which may result in frequent network disconnections. On the other hand, the network node density is usually extremely high in downtown areas during rush hours. Thus the correct approach for data dissemination in VANETs needs to consider the network diversity and adapt the dissemination mechanism according to the different network environments. Unfortunately the above traditional data dissemination mechanisms are mainly designed for general mobile ad hoc networks (MANETs) and sensor networks. In such networks, the topology and mobility are implicitly assumed to be homogeneous, i.e., the network does not usually contain mixed mobility and topology patterns. So those deliberately designed schemes, although work well in their intended environments, are not enough to cope with the diversity in VANETs.

Because of the diversity, there is no simple “one-for-all” solution for disseminating data to all recipients spreading across the metropolitan. Depending on where a target vehicle is located and how far away it is from the roadside unit
where the service resides, the issues to be addressed are quite different. In order to disseminate data to a far away vehicle, the data need to be forwarded over multi-hops, spanning a large geographic area. The forwarding path may go over some regions with low vehicles density, where a forwarding node is hard to find. So the end-to-end connection over large distance does not always exist. How to ensure the data delivery in the presence of disconnections becomes the most prominent problem. When the target vehicle moves closer to the roadside unit and is located in a vehicle densely populated area, disconnection is less of a concern. Instead, it becomes the major problem that many vehicles close to one another are requesting data at the same time, competing the shared wireless media. Therefore, how to efficiently make use of the limited bandwidth becomes the key issue. When a vehicle moves within the one-hop range of the roadside units, data can be delivered to the vehicle at the highest throughput. However, the vehicle only stays in the one-hop coverage of the roadside unit for very short time when it is moving. Thus, as a vehicle passes by the roadside unit, it is most desirable to extend the connection time between the vehicle and the roadside unit so as to disseminate more data. For more efficient data dissemination, multiple roadside units can be wirelessly connected together to form an infrastructure mesh network and cooperatively disseminate data to vehicles. Then how to distribute the data over the mesh nodes is an important issue.

In addition to diversity, several other unique properties further complicate the data dissemination in VANETs, which have never been considered in other types of ad hoc networks. Vehicles generally move at much higher rates of speed than nodes in other kinds of MANETs, and the link topology changes rapidly [22, 23]. As a result, many well studied structures for efficient data dissemination, such as tree, clustering, grid, are extremely hard to set up and maintain. Because of network diversity in both topology and mobility, the network can be sparse at some regions, which may result in frequent network disconnections and network partitions. In other areas where network node density is high, the conventional broadcast based mechanism for data dissemination may lead to broadcast storm [24]. Also, the widely used mobility models such as Waypoint model and random walk model no longer capture the mobility characteristics in VANETs, because vehicles have distinct controlled mobility patterns that are constrained to roadways.
layout and subject to vehicular traffic regulations. In urban areas, roads are often separated by buildings and other obstacles which can block radio communication, so communication is often restricted along roads.

The above challenges are unique to VANETs, particularly in large scale metropolitan. They have never been considered in designing data dissemination protocols for other forms of MANETs before, making the existing protocols insufficient to serve VANETs. It has been shown in [25, 26, 27, 28] that many traditional MANET data dissemination schemes result in poor overall network performance in VANETs, such as long delay, low data delivery rate, low bandwidth utilization and high network overhead.

1.3 Focus of This Dissertation

The goal of this dissertation is to provide a comprehensive solution to disseminate data from roadside units to moving vehicles at various locations of a metropolitan wide VANET. Starting from vehicles far away from the roadside units to vehicles within one-hop range of the roadside units, we develop different techniques for data dissemination. Our techniques will consider the identified challenges and form a cohesive data dissemination framework for VANETs. We briefly explain these techniques in the following four sections.

1.3.1 Data Access over Large Scale Sparsely Connected VANETs

A vehicle far away from the service roadside unit may have to go through multiple hops over long distance to access the data on the roadside unit. However, data access through multihop is much more difficult since VANET is highly mobile and sometimes sparse. The network density is related to the vehicle traffic density, which is affected by both location and time. Although it is very difficult to find an end-to-end connection for a sparsely connected network, the high mobility of vehicular networks introduces opportunities for mobile vehicles to connect with each other intermittently during moving.

To support data dissemination in the presence of disconnections in sparse
VANETs, we develop a *vehicle-assisted data delivery (VADD)* scheme. The scheme adopts the idea of *store, carry and forward*, where nodes carry the packet when routes do not exist, and forward the packet to the new receiver that moves into its vicinity. Different from existing carry and forward solutions, VADD makes use of the predictable vehicle mobility, which is limited by the traffic pattern and the road layout. The data forwarding path computed by VADD takes densely populated roads with high probability, which makes a suitable approach for forwarding data over sparsely connected VANETs. Therefore, even without end-to-end connection, the data packet can still be efficiently routed between vehicles and the roadside unit with high success rate and reasonable delay.

### 1.3.2 Data Pouring and Buffering in Densely Populated VANETs

In VANET, it is more effective to deploy roadside units in highly populated areas so as to serve more travelers with less cost. In these areas, data (such as traffic information or advertisement) need to be delivered to many vehicles and push-based approach is most widely used. The communication range of the roadside unit is limited but the data may need to be disseminated several miles along the road. Existing solutions are based on opportunistic data dissemination similar to gossip [29, 30]. Every vehicle buffers the data it receives, and relies on the intermittent connection between vehicles to propagate the data. However, the performance of the opportunistic dissemination suffers in the areas with high vehicle density due to MAC layer collisions.

To mitigate the excessive transmissions and congestion, we rely on the use of the partially predictable vehicle mobility limited by the road layout. Instead of spreading data throughout the network, the data are only broadcast along one or several selected main roads, called *axis roads*. The data are not actively pushed to roads that intersect with the axis roads, since vehicles on these roads will eventually reach a point intersecting with axis roads, where they can get the data.

We propose a multi-hop data pouring scheme, in which data are periodically broadcasted in a reliable way from data server to vehicles along the axis roads. To improve the performance, we propose a data buffering scheme to deploy fixed
roadside units at the intersections along the axis road to buffer the data poured from the source. Then they periodically rebroadcast the received data to serve the passing-by vehicles. By keeping a data copy at the intersection, the data dissemination can use different broadcast frequencies at different intersections. The data center only broadcasts the data for data invalidation or refresh the lost data copy. Our approach can greatly reduce the traffic injected into the network and increase the channel utilization. Thus more useful data can be delivered to vehicles.

1.3.3 Extending The Coverage of Roadside Unit

When a vehicle moves into the communication range of a roadside unit, data can be disseminated between them over one-hop link. In VANET, vehicles move fast and may only stay within the communication range of the roadside unit for a very short period of time. When a vehicle passes by a roadside unit, it can receive high signal strength when it is close to the roadside unit, but experiences poor link quality when entering or exiting the roadside unit coverage area. One hop link can provide the best throughput for delivering data to vehicles. However, as a vehicle spends a large portion of the connection time in the poor link quality area, the data throughput can be significantly reduced. This problem has been identified in several works [31, 32, 33], but a viable solution has yet to be proposed.

To address this problem, we focus on designing a distributed vehicle-to-vehicle relay solution which extends the service range of roadside units and allows drive-thru vehicles to maintain high throughput within an extended range. In order to simplify the deployment of our solution and ensure the compatibility with the existing 802.11 infrastructure, our solution is designed to be purely client-based, without any modification to existing 802.11 access points.

1.3.4 Cooperative Caching in Vehicular Infrastructure Mesh Networks

A single roadside unit can only provide service in a very limited geographic area. With recent advances in wireless mesh networking, multiple roadside units can be wirelessly connected to provide broader coverage. This model is called vehicular
*infrastructure mesh network.* In such a mesh network, the participating roadside units act as mesh routers (or mesh nodes), which perform routing and configuration functionalities among themselves. Vehicles can access the data by connecting to any mesh node and those mesh nodes provide backbone for disseminating data to vehicles. A mesh node does not necessarily have to store the data locally to serve vehicles. When a vehicle requests data from one mesh node, the data can be routed from the source to the requested mesh node over the mesh backbone, and then is delivered to the vehicle through the link between the vehicle and the mesh node. Typically, two types of radios are used in the mesh nodes: one for backbone communication among mesh nodes and the other for short communication between vehicles and mesh nodes. The mesh backbone communication is usually established using long range communication techniques.

Because of interest locality, many vehicles close to one roadside unit often share common interests, and request similar data. If one vehicle has accessed a data item from the roadside unit, it is quite possible that nearby vehicles access the same data item some time later. It will save a large amount bandwidth and time if later accesses to the same data are served by the nearby mesh nodes who has the data instead of the faraway data source. The concept of *cooperative cache*, which allows the sharing and coordination of cached data among multiple nodes, has been already applied to improve the system performance in wireless ad hoc networks. However, all the previous studies are at a very high level, leaving many design and implementation issues unanswered.

We focus on design and implementation of cooperative cache in vehicular infrastructure mesh networks. We propose solutions to find the best mesh nodes to cache the data based on the aggregated vehicle data access pattern on each mesh node. We propose an asymmetric cooperative cache approach, where the data requests are transmitted to the cache layer on every node, but the data replies are only transmitted to the cache layer at the intermediate nodes that need to cache the data. This solution not only reduces the overhead of copying data between the user space and the kernel space, it also allows data pipelines to reduce the end-to-end delay. We also study the effects of different MAC layers such as 802.11 based mesh networks and multi-interface multi-channel based mesh networks, on the performance of cooperative cache.
1.4 Organization

The remaining of the thesis is organized as follows. Chapter 2 reviews previous work on different aspects of data dissemination issues in VANETs. Chapter 3 presents our strategies to handle disconnections when disseminating data over sparsely connected VANETs. Chapter 4 focuses on improving the capacity and reliability when disseminating data in densely populated area. Chapter 5 presents a relay based scheme to extend the coverage of roadside units. In Chapter 6, we address the design and implementation issues for cooperative cache in vehicular infrastructure mesh nodes. We conclude the dissertation and discuss the future work in Chapter 7.
Related Work

Quite a few researches have been done on intervehicle communication so far. This section reviews the current state of art of VANET in a bottom-up approach.

2.1 MAC Layer

Similar as in all the other wireless systems, every communication unit in VANET is able to send and receive data packets into/from a shared medium in the air. Based on the FCC mandated band plan for dedicated short range communication (DSRC) [1, 5], the frequency spectrum will be divided in seven different channels, one control channel and six service channels. The control channel is regularly monitored by all vehicles.

Xu et al. [34] explore the feasibility of broadcasting safety messages between vehicles using the DSRC control channel and evaluate whether the QoS requirement of VANET safety applications (i.e. reasonably high transmission reliability and low latency) can be satisfied. Since the broadcasting is not inherently reliable, they have proposed two schemes to enhance the reliability: repetitive transmission and carrier sensing. Their results show the DSRC control channel is feasible for broadcasting safety message given the traffic pattern offered in the paper. However, the repetitive transmission scheme they proposed is not efficient, does not guaranty the reliability either. Later, Yadumurthy et al. [35] extended this work and further studied the reliable broadcasting problem. They adapted the Batch Mode Multicast MAC (BMMM) protocol originally proposed in [36] to improve
both reliability and efficiency for the broadcasting. In order to mitigate the channel competition and further reduce the delay to deliver critical emergency warning, slot-reservation MAC protocols are introduced in [37, 38].

The channel capacity available for broadcast is limited. If too many nodes are generating broadcast messages, there may be excessive channel contention. Artimy et al. [39] proposed to control the communication range by adjusting the transmission power, so that they can mitigate the excessive channel contention introduced by high density conditions to increase network capacity. Data dissemination in VANET does not limited to one hop transmission. Data delivery through multi-hops is also required by many applications. Korkmaz et al. [40] studied how to revise the existing IEEE 802.11 MAC protocols to better support multi-hop data dissemination in VANET. They proposed to employ link layer acknowledge mechanisms to improve the reliability of the multi-hop broadcast, while assigning the duty of forwarding and acknowledging the broadcast packet to only one vehicle to reduce the broadcast storm problem. MAC layer fairness problem has also been studied in [41, 42].

2.2 Network Layer

Maihöfer et al. [43] designed the framework to deliver messages to all vehicles inside a pre-specified destination region, which is useful for many location-based services. They proposed two-phase based message delivery: In the first phase, the message is unicasted to one specific target in the destination region by some geographic routing protocols, while in the second phase, the message is spread in the destination region. Xu et al. [44] proposed a opportunistic data dissemination scheme, which is similar to gossip [30, 45, 46]. In this approach, the data source periodically broadcasts some data which will be received and stored by the passing by vehicles. Whenever two vehicles move into the transmission range of each other, they exchange data. This approach does not rely on any infrastructure, and hence suitable for highly dynamic VANETs. However, the performance of the opportunistic dissemination scheme is poor in areas with high vehicle density due to MAC layer collisions [47]. This can easily lead to severe congestion and significantly reduce the data delivery ratio.
2.3 Application Layer

Some researchers also addressed application issues. Dikaiakos et al. [4] proposed a vehicular information transfer protocol to support location-oriented services by taking advantage of the VANET infrastructure. Huang et al. [7] designed a radio dispatch system for taxi drivers based on VANET. In TrafficView, which is part of the e-road project, Nadeem et al. [48] defined a framework to disseminate and collect information about the vehicles on the road. Transportation safety issues have been addressed in [6, 27], where vehicles communicate with each other and with the static roadside devices such as traffic light, bus shelters, and traffic cameras. Other applications such as real time video streaming between vehicles have been studied in [28].

2.4 Internetworking with Roadside Units

At the initial stage, since the market penetration rate is low and not many vehicles have the necessary wireless communication device installed, it is difficult to form a connected VANET only based on vehicle to vehicle communications. However, the manufacture/business owner can deploy roadside units to disseminate useful information. In this way, they can encourage more people to install wireless access devices in their vehicles. Hence, the data dissemination between vehicles and roadside units is very important at the initial stage. Currently there are several research groups working on this subject.

Bychkovsky et al. [33] performed an experiment to study the performance and the technical issues for moving vehicles to access WiFi access points deployed in metropolitan areas. In the experiment, they used nine cars to drive around Boston metropolitan area. Each car was equipped with a wireless enabled computer to collect data about AP access during their owner’s normal driving. The APs being accessed are residential “open” Wi-Fi APs which were not an engineered network. The wireless communication between the car and the access point was based on 802.11 technology. Through the measurement of over 290 “drive hours”, they had many interesting discoveries. In the experiment, their cars were able to successfully associate with APs and transfer data at the urban speed range from 0 to
60 km/hour. Also the possibility of finding AP connections was equal across the speed range. However, the connection duration with a single access point was quite limited (less than 60 seconds), and particularly short when the cars were moving at a high speed. Their experiment showed that the connection setup delay took several seconds, which is a significant issue considering the limited connection duration. They broke down the delay into several steps of the connection setup, i.e. scanning, association and IP acquisition with DHCP. The result showed that the DHCP delay accounted for the majority of the connection setup delay. This experiment verified the feasibility of accessing roadside access points from moving cars regardless of the speed. However, the biggest challenge was that the connection time may be quite short, so how to improve the vehicle to roadside connectivity and efficiently use the connection became the most urgent issues to solve.

The MobiSteer project [49] proposed a solution based on directional antennas and beam steering techniques to improve the vehicle to AP connectivity. The MobiSteer used a steerable beam directional antenna with a WiFi (802.11b/g) client node mounted on a moving car. A MobiSteer car first collected the information about the link quality of all accessible APs it met during previous traveling. This information was locally stored to build a database. One could tell from the database the link quality of an AP when the car was located at a specific location. Given this database, a MobiSteer car could select the best AP and directional antenna beam combination at any location along the drive, such that the throughput could be maximized. The MobiSteer approach was implemented and tested in filed experiment. The results showed that MobiSteer approach outperformed the traditional AP access using omni-directional antenna in AP connectivity duration and transmission data rate.

Given the vehicle to roadside communication capability, the MIT CarTel project [50] built a system to collect data from sensors located on vehicles and deliver data through vehicle to roadside AP links. A CarTel vehicle was equipped with an embedded computer coupled to a set of sensors. Those sensors could measure everything from engine processes to outside environment. Therefore, a CarTel vehicle could gather traffic-related data like speed, location, and direction of travel, weather conditions and related vehicle information, such as windshield-wiper, headlight use, and traction-control-system data. These data could be stored locally in the vehi-
cle and sent to roadside APs, which were eventually delivered to a central portal through a backbone network infrastructure. These data is stored in a centralized database for further analysis and visualization. CarTel has been deployed on six cars, running on a small scale in Boston and Seattle for over a year. It has been tested to analyze commute times, analyze metropolitan Wi-Fi deployments, and for automotive diagnostics.

So far, none of the aforementioned research focuses on providing a comprehensive data dissemination solution for VANET, which is the primary goal of this proposal.
Chapter 3

Data Dissemination over Large Scale Sparsely Connected VANETs

3.1 Introduction

Multi-hop data dissemination in VANET is envisioned to be useful in many applications. For example, without Internet connection, a moving vehicle may want to query a data center several miles away through a VANET. To further motivate our work, consider the widely deployed Wireless LANs or infostations [51] [9] which can be used to deliver advertisements and announcements such as sale information or remaining stocks at a department store; the available parking lot at a parking place; the meeting schedule at a conference room. Since the broadcast range is limited, only clients around the access point can directly receive the data. However, these data may be beneficial for people in moving vehicles which are far away. For example, people driving to shopping may want to query several department stores to decide where to go; a driver may query the traffic cameras or parking lot information to make a better road plan. All these queries may be issued miles or tens of miles away from the broadcast site. With a VANET, the requester can send the query to the broadcast site and get reply from it. In the above applications, the users can tolerate up to seconds or minute of delay as long as the reply eventually returns.

Although aforementioned services can be supported by the wireless infrastruc-
ture (e.g., 3G), the cost of doing this is high and may not be possible when such an infrastructure does not exist or is damaged. From the service provider point of view, setting up a wireless LAN is very cheap, but the cost of connecting it to the Internet or the wireless infrastructure is high. From the user point of view, the cost of accessing data through the wireless carrier is still high and most of the cellular phone users are limited to voice service. Moreover, in case of disaster, the wireless infrastructure may be damaged, whereas wireless LANs and vehicular networks can be used to provide important traffic, rescue and evacuation information to the users.

Multi-hop data delivery through VANETs is complicated by the fact that vehicular networks are highly mobile and sometimes sparse. The network density is related to the traffic density, which is affected by the location and time. For example, the traffic density is low in rural areas and during night, but very high in the large populated area and during rush hours. Although it is very difficult to find an end-to-end connection for a sparsely connected network, the high mobility of vehicular networks introduces opportunities for mobile vehicles to connect with each other intermittently during moving. Namboodiri et al. [22] showed that there is a high chance for moving vehicles to set up a short path with few hops in a highway model. Further, a moving vehicle can carry the packet and forward it to the next vehicle. Through relays, carry and forward, the message can be delivered to the destination without an end-to-end connection for delay-tolerant applications.

This chapter studies the problem of efficient data delivery in vehicular ad hoc networks. Specifically, when a vehicle issues a delay tolerant data query to some fixed site, we propose techniques to efficiently route the packet to that site, and receive the reply within reasonable delay. The proposed vehicle-assisted data delivery (VADD) is based on the idea of carry and forward [29], where nodes carry the packet when routes do not exist, and forward the packet to the new receiver that moves into its vicinity. Different from existing carry and forwarding approaches [29, 52, 53] it makes use of the predictable mobility in VANET, which is limited by the traffic pattern and the road layout. Extensive experiments are used to evaluate the proposed data delivery protocols. Results show that the proposed VADD protocols outperform existing solutions in terms of packet delivery ratio, data packet delay and protocol overhead.
The rest of this chapter is organized as follows. Section 3.2 describes how to model the data delivery delay. The VADD protocols will be presented in Section 3.3. Section 3.4 evaluates the performance of the proposed protocols. Section 6.5 concludes the chapter.

3.2 The VADD Model

In this section, we first give the assumptions, the overview of Vehicle-Assisted Data Delivery (VADD), and then present the VADD delay model.

3.2.1 Assumptions

We assume vehicles communicate with each other through short range wireless channel (100m-250m). The packet delivery information such as source id, source location, packet generation time, destination location, expiration time, etc, is specified by the data source and placed in the packet header. A vehicle knows its location by triangulation or through GPS device, which is already popular in new cars and will be common in the future. Vehicles enclose their own physical location, moving velocity and direction information in their periodic beacon messages, and these information can be overheard by their one-hop neighbors.

We assume that vehicles are equipped with pre-loaded digital maps, which provide street-level map and traffic statistics such as traffic density, vehicle speed on roads at different times of the day, and traffic signal schedule (e.g. the length of red signal interval) at intersections. Such kind of digital map has already been commercialized. The latest one is developed by MapMechanics [54], which includes road speed data and an indication of the relative density of vehicles on each road. Yahoo is also working on integrating traffic statistics in its new version of Yahoo Maps, where real traffic reports of major US cities are available. We expect that more detailed traffic statistics will be integrated into digital map in the near future. Note that the cost of setting up such a vehicular network can be justified by its application to many road safety and commercial applications [34, 6, 27], which are not limited to the proposed delay tolerant data delivery applications.
3.2.2 VADD overview

VADD is based on the idea of carry and forward. The most important issue is to select a forwarding path with the smallest packet delivery delay. Although geographical forwarding approaches such as GPSR [55] which always chooses the next hop closer to the destination, are very efficient for data delivery in ad hoc networks, they may not be suitable for sparsely connected vehicular networks.

As shown in Figure 3.1, suppose a driver approaches intersection $I_a$ and sends a request to the coffee shop (to make a reservation) at the corner of intersection $I_b$. To forward the request through $I_a \rightarrow I_c, I_c \rightarrow I_d, I_d \rightarrow I_b$ would be faster than through $I_a \rightarrow I_b$, even though the latter provides geographically shortest possible path. The reason is that in case of disconnection, the packet has to be carried by the vehicle, whose moving speed is significantly slower than the wireless communication.

![Figure 3.1. Find a path to the coffee shop](image)

![Figure 3.2. The transition modes in VADD](image)

In sparsely connected networks, vehicles should try to make use of the wireless communication channel, and resort to vehicles with faster speed otherwise. Thus, our VADD follows the following basic principles:

1. Transmit through wireless channels as much as possible.

2. If the packet has to be carried through certain roads, the road with higher speed should be chosen.

3. Due to the unpredictable nature of vehicular ad-hoc networks, we cannot expect the packet to be successfully routed along the pre-computed optimal
path, so dynamic path selection should continuously be executed throughout the packet forwarding process.

As shown in Figure 3.2, VADD has three packet modes: Intersection, Straight-Way, and Destination based on the location of the packet carrier (i.e., the vehicle that carries the packet.) By switching between these packet modes, the packet carrier takes the best packet forwarding path. Among the three modes, the Intersection mode is the most critical and complicated one, since vehicles have more choices at the intersection.

### 3.2.3 The VADD Delay Model

To formally define the packet delivery delay, we need the following notations.

- $r_{ij}$: the road from $I_i$ to $I_j$.
- $l_{ij}$: the Euclidean distance of $r_{ij}$.
- $\rho_{ij}$: the vehicle density on $r_{ij}$.
- $v_{ij}$: the average vehicle velocity on $r_{ij}$.
- $d_{ij}$: the expected packet forwarding delay from $I_i$ to $I_j$.

We assume the inter-vehicle distances follow Exponential distribution with mean distance equal to $1/\rho_{ij}$ [56]. Thus,

$$d_{ij} = (1 - e^{-R\rho_{ij}}) \cdot \frac{l_{ij} \cdot c}{R} + e^{-R\rho_{ij}} \cdot \frac{l_{ij}}{v_{ij}}$$  \hspace{1cm} (3.1)

where $R$ is the wireless transmission range, and $c$ is average one hop packet transmission delay. Equation 3.1 indicates that the inter-vehicle distances are smaller than $R$ on a portion of $1 - e^{-R\rho_{ij}}$ of the road, where wireless transmission is used to forward the packet. On the rest of the road, vehicles are used to carry the data. Apparently, larger traffic density make less portion completed by vehicle motion.

One way to view the VADD delay model is to represent the vehicular network as a directed graph, in which nodes represent intersections and edges represent the roads connecting adjacent intersections. The direction of each edge is the traffic
direction. The packet forwarding delay between two adjacent intersections is the weight of the edge. Given the weight on each edge, a naive optimal forwarding path selection scheme is to compute the shortest path from source to destination by applying Dijkstra’s algorithm. However, this simple solution does not work, since we cannot freely select the outgoing edge to forward the packet at an intersection. Only those edges with vehicles on it to carry packets can be the candidate path for packet forwarding. However we can not know for sure which direction the packet will go at the next intersection. In other words, it is impossible to compute the complete packet forwarding path.

To address this problem, we propose a stochastic model to estimate the data delivery delay, which is used to select the next road (intersection). We first introduce the following notations:

- \( D_{ij} \): The expected packet delivery delay from \( I_i \) to the destination if the packet carrier at \( I_i \) chooses to deliver the packet following road \( r_{ij} \).
- \( P_{ij} \): the probability that the packet is forwarded through road \( r_{ij} \) at \( I_i \).
- \( N(j) \): the set of neighboring intersections of \( I_j \).

As shown in Figure 3.3, for a packet at \( I_m \), the expected delay of delivering the packet through road \( r_{mn} \) is:

\[
D_{mn} = d_{mn} + \sum_{j \in N(m)} (P_{nj} \times D_{nj})
\] (3.2)
Figure 3.4 illustrates how to apply Equation 3.2 to a simple triangle road, which only contains three intersections $I_a$, $I_b$, and $I_c$. Suppose a data packet reaches $I_a$, and the destination is $I_c$. The forwarding scheme needs to decide whether to forward the packet through the road to $I_c$ or $I_b$. This is done by computing the value of $D_{ac}$ and $D_{ab}$, and choosing the smaller one. By applying Equation 3.2, we have the following linear equations:

\[
\begin{aligned}
D_{ac} &= d_{ac} \\
D_{ab} &= d_{ab} + P_{ba} \cdot D_{ba} + P_{bc} \cdot D_{bc} \\
D_{ba} &= d_{ba} + P_{ab} \cdot D_{ab} + P_{ac} \cdot D_{ac} \\
D_{bc} &= d_{bc} \\
D_{cb} &= 0 \\
D_{ca} &= 0
\end{aligned}
\] (3.3)

Note that both $d_{cb}$ and $d_{ca}$ are equal to 0, since the packet already arrives at destination $I_c$, and will not be forwarded anymore. We can easily solve Equation 3.3 and get $D_{ac}$ and $D_{ab}$:

\[
\begin{align*}
D_{ac} &= d_{ac} \\
D_{ab} &= \frac{1}{1 - P_{ab} \cdot P_{ba}} \times (d_{ab} + P_{ba} \cdot d_{ba} + P_{ba} \cdot P_{ac} \cdot d_{ac} + P_{bc} \cdot d_{bc})
\end{align*}
\]

Unfortunately, to find the minimum forwarding delay between two arbitrary intersections is impossible, since it involves unlimited unknown intersections. How-
ever, by placing a boundary including the source and the destination in a connected graph, we are able to find the expected minimum forwarding delay between them. Figure 3.5 shows one such boundary which includes the sender and the destination (hot spot). The boundary used in this paper is a circle with its center point at the destination. The radius of the boundary circle is 4000 meters if the distance between the packet and the destination is less than 3000 meters; otherwise, the radius is the distance between the packet and the destination plus 1000 meters. Certainly there are many other ways to place the boundary, as long as the destination is enclosed. Since only the roads within the boundary are used as available paths to compute the delay, a large boundary covering more high-density streets can generally find more close-to-optimal paths, but with more computation overhead. Thus, there is a tradeoff between computational complexity and accuracy in delay estimation when selecting the boundary. Since this is not the major concern of this dissertation and it does not affect the correctness of our algorithms, we will not further discuss it in this dissertation.

Since the number of intersections inside the boundary is finite, we can derive Equation 3.2 for each outgoing edge of every intersection within the boundary (similar to the method used to derive Equation 3.3). In this way, an $n \times n$ linear equation system is generated, where $n$ is the number of roads within the boundary.

To follow the general representation of linear equation systems, we rename the unknown $D_{ij}$ as $x_{ij}$, rename the subscript $ij$ of $d_{ij}$ and $x_{ij}$ with a unique number for each pair $ij$, and rename the subscript of $P_{ij}$ by its position in the equations.
Then, we can derive \( n \) linear equations with \( n \) unknowns \( x_1, x_2, \cdots, x_n \):

\[
\begin{align*}
  x_1 &= d_1 + P_{11}x_1 + P_{12}x_2 + \cdots + P_{1n}x_n \\
  x_2 &= d_2 + P_{21}x_1 + P_{22}x_2 + \cdots + P_{2n}x_n \\
  &\vdots \\
  x_n &= d_n + P_{n1}x_1 + P_{n2}x_2 + \cdots + P_{nn}x_n
\end{align*}
\]

It can be easily transformed to the following matrix.

\[
(P_{11} - 1)x_1 + P_{12}x_2 + \cdots + P_{1n}x_n = -d_1 \\
P_{21}x_1 + (P_{22} - 1)x_2 + \cdots + P_{2n}x_n = -d_2 \\
\vdots \\
P_{n1}x_1 + P_{n2}x_2 + \cdots + (P_{nn} - 1)x_n = -d_n
\]

which is equivalent to

\[
(P - E) \cdot X = -D 
\]  

(3.4)

where

\[
P = \begin{bmatrix}
P_{11} & P_{12} & \cdots & P_{1n} \\
P_{21} & P_{22} & \cdots & P_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
P_{n1} & P_{n2} & \cdots & P_{nn}
\end{bmatrix}, \quad E = \begin{bmatrix}
1 & 0 & \cdots & 0 \\
0 & 1 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1
\end{bmatrix},
\]

\[X = \begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_n
\end{bmatrix} \text{ and } D = \begin{bmatrix}
d_1 \\
d_2 \\
\vdots \\
d_n
\end{bmatrix}\]

We can prove that this linear equation system has one unique solution (see Appendix). The typical way to solve this equation is to use the Gaussian Elimination algorithm, which is known to be solved in time \( \Theta(n^3) \).

By solving Equation 3.4, we get \( D_{ij} \) for the current intersection \( I_i \). The packet carrier can sort \( D_{ij} \) for each neighboring intersection \( I_j \), and forward the packet to the road with smaller delay. As a result, among all the vehicles within communication range (called contacts) available at the intersection, the packet will be
 forwarded to the one on the road with the smallest delay. If no contact is available or all available contacts are going through roads with longer delay than the packet carrier’s next traveling road, the packet carrier passes the intersection with the packet, and looks for the next forwarding opportunity.

3.3 Vehicle-Assisted Data Delivery Protocols

In this section, we present the VADD protocols. We first present the protocols used in the Intersection mode and then present the contact model and protocols on the Straightway.

3.3.1 VADD Protocols Used in the Intersection Mode

![Diagram](image)

*Figure 3.6. Select the next vehicle to forward the packet*

By deriving and solving Equation 3.4 at the intersection, the packet carrier can sort all the outgoing directions and check if there is a contact available to help forward through that direction. However, to determine the next hop among all available contacts and ensure a packet to go through the pre-computed direction is not trivial. As shown in Figure 3.6, vehicle A has a packet to forward to certain destination. Assume the optimal direction for this packet is North. There are two available contacts for the packet carrier: B moving south and C moving north. A has two choices on selecting the next hop for the packet: B or C. Both choices aim at forwarding the packet towards North: selecting B because B is geographically closer towards North and provides better possibility to exploit the wireless communication (e.g. B can immediately pass the packet to D, but C cannot;) whereas
selecting $C$ because $C$ is moving in the packet forwarding direction. These two choices lead to two different forwarding protocols: Location First Probe (L-VADD) and Direction First Probe (D-VADD).

3.3.1.1 Location First Probe (L-VADD)

Given the preferred forwarding direction of a packet, L-VADD tries to find the closest contact towards that direction as the next hop. First, based on Equation 3.4, $D_{ij}$ can be obtained for each outgoing road $r_{ij}$ at intersection $I_i$. As a result, each outgoing road is assigned a priority where smaller $D_{ij}$ has higher priority. Next, the packet carrier checks the outgoing directions starting from the highest priority. For a selected direction, the packet carrier chooses the next intersection towards the selected direction as the target intersection, and apply geographical greedy forwarding towards the target intersection to pass the packet. If the current packet carrier cannot find any contact to the target intersection, it chooses the direction with the next lower priority and re-starts the geographical greedy forwarding towards the new target intersection. This process continues until the selected direction has lower priority than the packet carrier’s current moving direction. At this time, the packet carrier will continue carrying the packet.

![Figure 3.7. A scenario of routing loop](image)

As shown in Figure 3.6, vehicle $A$ forwards the packet to $B$. Seems like this is better than selecting $C$ as the next hop, since $B$ can immediately forward packet to $D$. Even if $D$ does not exist, selecting $B$ seems as good as selecting $C$, since $B$ will meet $C$ shortly and the packet can be passed to $C$ anyway. However, L-VADD may result in routing loops. Figure 3.7 shows one such scenario. Assume the North direction has the highest priority and East has the second highest priority. $A$ first checks North and can not find any contact. Then, it checks East, and finds $B$
which is closer towards East. Thus, it forwards the packet to $B$. Upon receiving the packet, $B$ checks the North direction first and finds $A$ is closer towards North, and then passing the packet back to $A$. There is a loop between $A$ and $B$.

A simple solution to break the routing loop is to record the previous hop(s) information. As in the above example, $A$ records its own id as the previous hop before forwarding the packet to $B$. When $B$ receives the packet, and decides to forward the packet to $A$, it checks the previous hop record and finds that $A$ is the previous hop. To avoid a routing loop, $B$ will not forward the packet to $A$, and look for the next available contact.

A routing loop may involve $n(n > 2)$ nodes. To detect such a routing loop, all these previous $n$ hops should be recorded. However, such loop detection mechanism dramatically degrades the forwarding performance, since the detection mechanism may prevent many valid nodes from being considered as the next hop. As shown in Figure 3.7, if $A$ is the packet carrier after a routing loop has been detected, and there is no other contact available except $B$. Suppose after both $A$ and $B$ pass the center of the intersection, $A$ continues going East and $B$ to North. The packet should be forwarded to $B$ since $B$ will move towards the best direction, and the path between $A$ and $B$ becomes loop-free. However, as the packet records $B$ as the previous hop, forwarding the packet to $B$ is not allowed. Therefore, even though we can record previous hop information to detect routing loops, many valid forwarding paths cannot be used.

### 3.3.1.2 Direction First Probe (D-VADD)

Routing loop occurs because vehicles do not have an unanimous agreement on the order of the priority, and then do not have an agreement on who should carry the packet. To address this issue, D-VADD ensures that everyone agrees on the priority order by letting the vehicle moving towards the desired packet forwarding direction carry the packet.

In D-VADD, the direction selection process is the same as L-VADD. For a selected direction, instead of probing by location (in L-VADD), D-VADD selects the contacts moving towards the selected direction. Among the selected contacts, the one closest to the selected direction is chosen as the next hop. As shown in Figure 3.6, D-VADD selects $C$ as the next hop when the selected direction is
North. Since $B$ is not moving North, it will not be considered. Therefore, D-VADD only probes vehicles moving towards the direction whose priority is higher than or equal to the moving direction of current packet carrier. As the probing strictly follows the priority order of the direction, D-VADD has the following property: Any subsequent packet carrier moves towards the direction whose priority is higher than or equal to that of the current packet carrier.

**THEOREM 1.** *D-VADD is free from routing loops at intersection areas.*

*Proof.* By contradiction, suppose a routing loop occurs and node $A$ and $B$ are in the circle, which indicates that at least one packet forwarded from $A$ passes through $B$ and returns to $A$. Consider the first case that $A$ and $B$ are moving in the same direction, and the packet is forwarded from $A$ to $B$. It indicates that $B$ is closer towards the destination direction than $A$, while packet passing back to $A$ indicates the reverse. In the second case, if $A$ and $B$ move towards different direction, the packet forwarded from $A$ to $B$ indicates $B$ is moving towards the direction of higher priority than $A$’s, while the packet passing back to $A$ shows $A$’s direction has higher priority. Both cases lead to contradictions. Therefore, there is no routing loop in D-VADD.

3.3.1.3 Hybrid Probe (H-VADD)

Comparing to other VADD protocols, L-VADD without loop detection can minimize the packet forwarding distance and hence the delay if there is no loop. However, the routing loop in L-VADD severely affects the performance and leads to a low packet delivery ratio. Loop detection mechanism can remove the routing loop, but may also increase the forwarding delay. D-VADD is free from routing loops; however, they give priority to the moving direction and may suffer from long packet forwarding distance, and hence long packet delivery delay.

An ideal VADD protocol should minimize the geographic forwarding distance and does not have routing loops. To achieve this goal, we design a scheme called Hybrid Probe (H-VADD), which works as follows. Upon entering an intersection, H-VADD behaves like L-VADD with loop detection. If a routing loop is detected, it immediately switches to use D-VADD until it exits the current intersection. In this way, H-VADD inherits the advantage of using the shortest forwarding path in
L-VADD when there is no routing loop, and use D-VADD to address the routing loop problem of L-VADD.

3.3.1.4 The Problem of Disagreement and Redundant Computation

At an intersection, if the preferred forwarding direction of a packet is calculated at each hop of the forwarding nodes, the following two problems may occur. **Disagreement on preferred direction:** Each node independently derives and solves Equation 3.4 only based on the local information provided by their own digital maps. It is possible that two nodes do not have exactly the same traffic statistics (due to different map source, updating schedule and etc.). It is possible that two successive forwarding nodes obtain different expected forwarding delay for the same next road, so they may use different optimal directions to forward the packet. Then the packet may suffer from routing loops, similar to that in L-VADD. **Redundant computation:** In VADD, all the forwarding nodes within the same intersection area should follow exactly the same computation process, and ideally get the same preferred forwarding direction for a given packet. Thus, it may waste computation resources if multiple nodes do the computation several times.

The above two problems exist in all three VADD protocols: L-VADD, D-VADD, and H-VADD. To deal with these problems, only the first node in the intersection area receiving the packet performs the computation, and gets the priority order of the next forwarding direction/road for the packet. This information is enclosed in the packet header, and kept until the packet is forwarded out of the current intersection. The subsequent forwarding nodes in the same intersection do not repeat the computation. Instead, they check the packet header and forward the packet based on the computed priority order. In this way, only one computation is performed for a packet at one intersection, and the disagreement problem will be solved.

3.3.2 Calculating $P_{ij}$

In this section, we provide solutions to calculate $P_{ij}$ used in Section 3.2. Specifically, we choose D-VADD as the data delivery protocol, because of its simplicity in modeling the packet forwarding process. Certainly, other protocols such as
L-V ADD and H-V ADD can be modeled to calculate $P_{ij}$ in a similar way. The calculation of $P_{ij}$ under other VADD protocols should provide similar results since the different VADD protocols follow similar principle to find the optimal forwarding path through the roads with high vehicle density.

We focus on the normal traffic layout, where each road has one-way or two-way traffic and the intersections are either signalized or isolated [26]. Throughout this section we assume the vehicle arrivals at intersections follow Poisson distribution.

The expected time that a packet carrier stays in the Intersection Mode is referred to as the contacting time. The contacting time at a signalized intersection $I_i$, denoted as $t_i$, is only related to the length of the signal interval at $I_i$, and we assume it can be obtained from the digital map. In an isolated intersection, vehicles in all directions can smoothly go through without being stopped. For a vehicle at $I_i$, we assume the average vehicle speed going through the intersection is the same as the average vehicle speed at the outgoing road. Let $R_{int}$ denote the radius of the intersection area which is a circle with the intersection point as the center. Formula 3.5 computes the contacting time ($T_{ij}$) for packet carriers which enter intersection $I_i$, and move towards neighbor intersection $I_j$.

$$T_{ij} = \begin{cases} t_i, & I_i \text{ is signalized} \\ \frac{2R_{int}}{v_{ij}}, & I_i \text{ is isolated} \end{cases} \tag{3.5}$$

The packet carrier is able to forward the packet towards road $r_{ij}$ at $I_i$, only if it can meet at least one contact going towards road $r_{ij}$. Next, we calculate the probability ($CP_{ij}$) for a packet carrier to meet at least one contact towards road $r_{ij}$, when the carrier moves within the intersection area. Let $N(T_{ij})$ denote how many contacts moving towards road $r_{ij}$ can be seen in the intersection area within time interval $T_{ij}$, and let $\lambda_{ij}$ denote the average rate of contacts leaving $I_i$ and moving towards road $r_{ij}$, which can be computed as $\lambda_{ij} = \rho_{ij} \cdot v_{ij}$ ( $\rho_{ij}$ and $v_{ij}$ are defined in Section 3.2.3). According to the definition of Poisson distribution, \[ CP_{ij} = \text{Prob}(N(T_{ij}) \geq 1) = 1 - \text{Prob}(N(T_{ij}) = 0) = 1 - e^{-\lambda_{ij} T_{ij}} \frac{(\lambda_{ij} T_{ij})^0}{0!} \]
\[= 1 - e^{-\rho_{ij}v_{ij}T_{ij}}\]

The VADD protocols forward a packet towards the best possible direction at the intersection. If intersection \(I_i\) only has two outgoing roads \(r_{ia}\) and \(r_{ib}\) and satisfies \(D_{ia} < D_{ib}\) with contacting probability \(CP_{ia}\) for contacts towards road \(r_{ia}\) and \(CP_{ib}\) for contacts towards road \(r_{ib}\) respectively, \(P_{ia}\) would be equal to \(CP_{ia}\), and \(P_{ib}\) would be \(CP_{ib} - CP_{ia} \cdot CP_{ib}\). This is due to the reason that the path with the expected minimum delivery delay will be selected, if both contacts are available when the packet carrier passes the intersection \(I_i\). Therefore, to compute \(P_{ij}\) at \(I_i\), we need to first sort \(CP_{ij}\) for all \(j \in N(i)\) by the non-decreasing order of \(D_{ij}\). However, as \(D_{ij}\) cannot be obtained at this stage, we use the angle between the direction of road \(r_{ij}\) and the vector from the current intersection to the destination, denoted as \(\theta_{ij}\), to approximate \(D_{ij}\), because a road with smaller angle will more likely lead to a location closer to the destination. The sorted list of \(CP_{ij}\) looks like:

\[CP_{ij1}, CP_{ij2}, CP_{ij3}, \ldots, CP_{ijn}; \quad \text{where } n = |N(i)|\]

The subscripts of \(j_i\)s implicitly indicate a meaningful order:

\[\theta_{ij1} \leq \theta_{ij2} \leq \theta_{ij3} \leq \cdots \leq \theta_{ijn}\]  \((3.6)\)

By using basic probability, we can calculate the probability of a packet being forwarded to road \(r_{ij}\) at \(I_i\). This result is denoted as \(P'_{ij}\).

\[
P'_{ij1} = CP_{ij1}
\]
\[
P'_{ij2} = CP_{ij2} - CP_{ij1} \cdot CP_{ij2}
\]
\[
P'_{ij3} = CP_{ij3} - (CP_{ij1} \cdot CP_{ij3} + CP_{ij2} \cdot CP_{ij3})
\]
\[+ CP_{ij1} \cdot CP_{ij2} \cdot CP_{ij3}
\]
\[\vdots \]

Suppose the packet carrier will move to road \(r_{ijc}\) (either go straight or make a
turn) after passing $I_i$, the packet will only be forwarded to the road that has higher or equal priority. That is, for a road $r_{ij}$, if $k > c$, $P_{ij}$ equals to zero, since the carrier will continue to buffer data instead of forwarding it towards lower priority roads. Thus, under the condition that the packet carrier goes to road $r_{ij}$ after leaving $I_i$, the probability that road $r_{ijp}$ will be chosen as the packet forwarding direction can be defined as the following conditional probability:

$$P_{ijp|ijc} = \text{Prob}\{\text{packet forwarded to } r_{ijp} \mid \text{carrier goes to } r_{ijc}\}$$

and

$$P_{ijp|ijc} = \begin{cases} 
    P'_{ijp}, & \forall p < c \\
    1 - \sum_{s=1}^{c-1} P'_{ij_s}, & p = c \\
    0, & \forall p > c 
\end{cases} \quad (3.7)$$

Let $Q_{ic}$ denote the probability of a vehicle moving (going straight or turning) from the current intersection $I_i$ towards the next adjacent intersection $I_c$. $P_{ij}$ can be calculated by the following:

$$P_{ij} = \sum_{c \in N(i)} Q_{ic} \times P_{ijp|ijc} \quad (3.8)$$

The complexity of calculating $P_{ij}$ is dominated by the step of calculating $P'_{ij}$, and it is given by

$$\Theta(\sum_{k=1}^{N(i)} \binom{N(i)}{k}) = \Theta(2^{N(i)})$$

Since one intersection is only directly connected with several neighboring intersections in reality, $N(i)$ is bounded and fairly small, therefore, $2^{N(i)}$ can be seen as a constant. So the complexity of computing $P_{ij}$ for all $n$ roads inside the boundary is $\Theta(n)$.

### 3.3.3 Data Forwarding in StraightWay Mode and Destination Mode

Data forwarding in the StraightWay mode is much simpler than the intersection mode, since the traffic is at most bi-direction. We can simply specify the inter-
section ahead, which is jointed by the current road, as the target, and then apply GPSR [55] towards the target location. If there is no vehicle available to forward ahead, the current packet carrier continues to carry the packet. Certainly, there may be better solutions. For example, when the packet carrier meets a vehicle in the opposite direction, the estimated delay from the current vehicle position may be different when the vehicle received the packet. As a result, the packet carrier may decide to take the intersection behind as the target location. However, checking such cases may increase the computation overhead and the chance of such cases may be small. Due to space limit, we will leave these optimizations as future work.

A packet switches to the Destination Mode when its distance to the destination is below a predefined threshold. The location of the destination becomes the target location, and GPSR is used to deliver the packet to the final destination.

### 3.4 Performance Evaluations

In this section, we evaluate the performance of several VADD protocols including L-VADD, D-VADD, and H-VADD. Since the L-VADD protocol may have routing loops, we evaluate two versions of them: L-VADD (with loop) and L-VADD (loop-free). It is shown in our simulation that almost all the intersection routing loops in L-VADD (with loop) can be detected by checking previous three-hop information, so L-VADD(loop-free) encloses previous three-hop information in every forwarding packet to avoid intersection routing loops. The H-VADD protocol is a hybrid of the L-VADD protocol and the D-VADD protocol. We compare the performance of the VADD protocols to several existing protocols: DSR protocol [10], the epidemic routing protocol [29] and GPSR [55]. Since GPSR is not proposed for sparsely connected networks, its performance is very poor in VANETs. To have a fair comparison, we extend GPSR by adding buffers. In this way, GPSR (with buffer) can be considered as a simple carry and forward protocol.

The experiment is based on a $4000m \times 3200m$ rectangle street area, which presents a grid layout. The street layout is derived and normalized from a snapshot of a real street map in Topologically Integrated Geographic Encoding and Referencing (TIGER) database [57] from U.S. Census Bureau. These map data
Table 3.1. Simulation Setup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation area</td>
<td>$4000m \times 3200m$</td>
</tr>
<tr>
<td># of intersections</td>
<td>24</td>
</tr>
<tr>
<td>Intersection area radius</td>
<td>200m</td>
</tr>
<tr>
<td>Number of vehicles</td>
<td>150, 210</td>
</tr>
<tr>
<td># of packet senders</td>
<td>15</td>
</tr>
<tr>
<td>Communication range</td>
<td>200m</td>
</tr>
<tr>
<td>Vehicle velocity</td>
<td>15 - 80 miles per hour</td>
</tr>
<tr>
<td>CBR rate</td>
<td>0.1 - 1 packet per second</td>
</tr>
<tr>
<td>Data packet size</td>
<td>10 B - 4 KB</td>
</tr>
<tr>
<td>Vehicle beacon interval</td>
<td>0.5 sec</td>
</tr>
<tr>
<td>Packet TTL</td>
<td>128 sec</td>
</tr>
</tbody>
</table>

are transformed into the data format that can be used by ns2, based on techniques presented in [58]. The MAC layer protocol follows 802.11 with DCF enabled.

The mobility pattern is generated similar to that of [58], but we need to model unevenly distributed traffic. We revised the software in [58] to first compute the traveling time on each road based on the length and speed limit of the road, and then let each vehicle select the shortest path to the destination. Thus, roads with high speed limit are chosen with higher probability, which generates uneven traffic density. The initial distribution follows the traffic density distribution of the original map (i.e. more crowded roads are deployed with relatively more vehicles and less interspace between vehicles). Then, each vehicle randomly chooses one of the intersection as its destination, and moves along the road to this destination. Immediately after it arrives the destination, the vehicle randomly selects another
intersection as the next destination and moves towards it. The TIGER database contains road type information for each road, and we assign the speed limit (20-75 miles per hour) to each road based on the road type information, for example, 20 mile/hour for unseparated downtown streets, and 75 miles/hour for highways. The vehicles follow the speed limit assigned to the road they are traveling on, with a variance of 5 miles per hour. For simplicity, we only consider the case of isolated intersection, and the node contacting time at an intersection is calculated by Equation 3.5. Figure 3.8 shows a snapshot of the simulation area.

Two fixed sites are deployed on the rightmost vertical road in Figure 4.4. Among all vehicles, 15 of them are randomly chosen to send CBR data packet to one of the fixed sites during the move. To evaluate the performance on different data transmission density, we vary the data sending rate (CBR rate) from 0.1 to 1 packet per second. All experiment parameters are shown in Table 6.4. In order to find out the direction to forward a packet to a given fixed site, the priority ranking of the outgoing roads at the intersections for that fixed site are pre-computed and loaded to the vehicle as the simulation starts. The performance of the protocols are measured by the data delivery ratio, the data delivery delay, and the generated traffic overhead.

### 3.4.1 The Data Delivery Ratio

![Data delivery ratio as a function of the data sending rate](image)

Figure 3.9. Data delivery ratio as a function of the data sending rate

In this section, we compare the performance of VADD protocols with epidemic
routing, GPSR (with buffer), and DSR in terms of data delivery ratio, and examine how it is affected by the data transmission density and the vehicle density.

Figure 3.9 shows the data delivery ratio as a function of the data sending rate and compares the performance under different vehicle density settings. As shown in the figure, DSR has the lowest data delivery ratio and is not suitable for sparsely connected vehicular networks. Although GPSR (with buffer) is implemented in a carry and forward way, it is not a good choice since the geographical approach sometimes leads to void areas with few vehicles passing by, and it cannot make use of the traffic patterns. Therefore, its delivery ratio is poor when the vehicle density is low, as shown in Figure 3.9(a). However, when vehicle density is high (in Figure 3.9(b)), where the connectivity is much better than the previous scenario, GPSR achieves very good delivery ratio, since the node mobility will help carry and forward the packets which temporarily reach the void zone. Intuitively, epidemic routing explores every possible path to the destination, and should represent the upper bound of the data delivery ratio. This is true when the data sending rate is low (e.g., when the data rate is 0.1 packet per second), and the node density is low. However, as the data sending rate increases, the epidemic routing protocol underperforms most VADD protocols. This is due to MAC layer collisions. As the number of data requests increases, the network traffic dramatically increases in epidemic routing (see Figure 3.12), thus increasing the number of collisions and reducing the packet delivery ratio. At more densely deployed network as Figure 3.9(b), the delivery ratio of the epidemic protocol drops even faster. While the epidemic routing is very sensitive to the data rate and nodes density, the VADD protocols, particularly H-VADD, steadily hold the close-to-optimum delivery ratio at different settings.

Figure 3.9 also compares several VADD protocols. Among them, the H-VADD protocol has the benefits of both L-VADD and D-VADD, presenting the best delivery ratio. As discussed in the previous section, loop detection prevents some packets from being sent to the loop vulnerable neighbors, which reduces the chance of using some valid good paths. However, with a high vehicle density, intersection routing loops do not occur frequently, and the L-VADD (loop-free) protocol does not need to exclude too many innocent nodes to recover from the loop, and its delivery ratio becomes higher.
Figure 3.10. Percent of data packets dropped due to routing loops or MAC layer packet collisions (150 nodes)

The L-VADD (with loop) protocol has the lowest data delivery ratio among the VADD protocols, and performs especially poor when the node density is low, since routing loops frequently happen and lead to packet drops. Figure 3.10 compares the percentage of the data packet dropped due to TTL or MAC layer collision at a 150-node setting. As can be seen from the figure, three VADD protocols (L-VADD, D-VADD, and H-VADD) have similar percentage of packet drops. Compared to these VADD protocols, the L-VADD (with loop) protocol has a much higher packet drop rate; i.e., about 5 times higher. Figure 3.10 also verifies the effectiveness of the routing loop detection mechanism used by the loop-free L-VADD protocol.

From the figure, we can also see that the dropping rate of the L-VADD (with loop) protocol is reduced as the data sending rate increases. The is because most packets are dropped due to routing loops instead of congestion using the 150-node setting. Routing loops only occur at some particular time intervals. When the data sending rate is high, more packets are buffered and delivered before a routing loop occurs. Since the number of dropped packets due to routing loops does not change too much, but the total number of delivered packets increases as the data sending rate increases, the percentage of data packet drops becomes lower when the data sending rate increases.
3.4.2 The Data Delivery Delay

In this section we compare the data delivery delay from moving vehicles to fixed sites using carry and forward schemes. Here, we do not consider DSR since its data delivery ratio is too low. Similarly, we do not consider the L-VADD protocol due to its low delivery ratio compared to the D-VADD protocol. Note that a low delivery ratio may reduce the average data delivery delay since most undelivered packets may experience long delay. This is especially true in the DSR protocol, which only forwards packets through wireless communication whereas other carry and forward protocols also rely on the vehicle movement.

Figure 4.10 shows the change of the data delivery delay by increasing the data sending rate. Epidemic routing presents the optimum delivery delay only when the data rate is very low. As the data sending rate increases, the delay of the epidemic routing scheme also increases, because epidemic routing generates many redundant packets. As the traffic load increases, many packets may be dropped. Even though the redundant copies can help deliver the packet, the delay increases. GPSR has
relatively low data delivery delay at low node density (Figure 4.10(a)), but it is not meaningful simply because of its low delivery ratio. A valid comparison is when the GPSR protocol, the epidemic routing protocol, and the VADD protocols have similar delivery ratio, e.g., at data rate below 0.4 in Figure 4.10(b). In this case, GPSR shows much longer delivery delay because it does not consider the vehicle traffic pattern when making decisions.

The H-VADD protocol presents similar delivery delay as the D-VADD protocol when the vehicle density is low, since it relies more on D-VADD for loop recovery because of more routing loops. When the vehicle density is high, the delay of the H-VADD protocol is lower than that of the D-VADD protocol, but close to that of the L-VADD protocol. This shows that it behaves more like the L-VADD protocol, but has better packet delivery ratio than the loop free L-VADD. These results verify that H-VADD effectively captures the advantages of both L-VADD and D-VADD.

The delivery delay is affected by the delivery ratio. Some extreme long-delay packets may greatly increase the mean value, and the average delivery delay generally becomes smaller when less packets are successfully delivered. So the delivery delay of H-VADD appears to be larger than than some other VADDs simply because it delivers more packets. To better study the delivery delay, we examine the “The lowest 75% delivery delay”, which is the average delay of the lowest 75% packets. As shown in Figure 4.10(c), the delay of H-VADD is only half of D-VADD. It is similar to L-VADD because it behaves more like L-VADD when the node density is high.

3.4.3 Data Traffic Overhead

In this section, we evaluate the overhead of the carry and forward protocols by using the number of data packets generated per second, which is a summation of individual packet-hops. For example, if a generated packet is forwarded 10 hops, the packet overhead is counted as 10 packet-hops. The control packets are not included. The reason is that the proposed VADD protocol is essentially a location-based routing protocol and it does not require any more control packets than other location-based routing protocols. All VADD protocols and GPSR require the same
number of control messages which are the beacon messages to report the node location. The control message overhead depends on the beacon interval, which is set to 0.5 sec for all the evaluated protocols. Thus, in VADD protocols and GPSR, each node generates the same amount of control traffic regardless of the data rate, topology and mobility. All results shown in this section are based on the 210-node deployment scenario. Figure 3.12 shows the generated packet overhead as a function of the data sending rate. As the data sending rate increases, the number of packets generated by all protocols also increases. However, the increasing trend is different. The overhead of epidemic routing increases much faster than other protocols due to the redundant packets generated.

For the VADD protocols, L-VADD (with loop) has the highest overhead due to loops whereas all the other VADD protocols have about the same low overhead. Compared to D-VADD,

3.4.4 The Impact of Data Packet Size

Figure 3.13 illustrates the impact of data packet size on the performance of the GPSR protocol, the epidemic routing protocol, and the VADD protocols. Since all the VADD protocols are affected by the data size in similar way, we choose H-VADD to represent the VADD protocols in the comparison. Larger packet size consumes more bandwidth and generates more contention for the limited wireless
As shown in Figure 3.13(a), the total injected data traffic using the epidemic protocol increases much faster than GPSR and H-VADD. We intentionally choose the setting at a very low data sending rate (0.1 per second), where the delay of the epidemic routing is close to H-VADD, and the delivery ratio is slightly better than H-VADD at the starting size (10 Bytes) due to the help of large amount of redundant packets. The delivery ratio of the epidemic routing protocol drops much faster than the H-VADD protocol as the data size increases (see Figure 3.13(b)). As shown in Figure 3.13(c), the delivery delay of the epidemic protocol increases dramatically as the packet size increases due to the congestion caused by the huge traffic load. The delay of the GPSR protocol slightly decreases as the packet size increases since some long delay packets are dropped. From the figure, we can also see that the H-VADD protocol has the lowest data delivery delay for different data sizes.

Figure 3.13. Impact of data packet size
3.5 Conclusions

With the ability of internetworking through wireless communication among moving vehicles, many data delivery applications can be supported without expensive infrastructure cost. However, existing protocols are not suitable for supporting delay tolerate applications in sparsely connected vehicular networks. To address this problem, we adopted the idea of carry and forward, where a moving vehicle carries the packet until a new vehicle moves into its vicinity and forwards the packet. Different from existing carry and forward solutions, we make use of the predicable vehicle mobility, which is limited by the traffic pattern and road layout. We proposed several vehicle-assisted data delivery (VADD) protocols: L-VADD, D-VADD, and H-VADD based on the techniques used for road selection at the intersection. Experimental results showed that the proposed VADD protocols outperform existing solutions in terms of packet delivery ratio, data packet delay and traffic overhead. Among the proposed VADD protocols, the H-VADD protocol has much better performance.
Data Pouring and Buffering in Densely Populated VANETs

4.1 Introduction

In VANET, it is more effective to deploy roadside units in highly populated areas so as to serve more travelers with less cost. In these areas, one data item (such as traffic information or advertisement) can be delivered to many vehicles to provide increased convenience and efficiency. For example, an alert message on traffic accident or traffic jam can be propagated tens of miles along the road to help drivers select a better road. Department stores can disseminate sale advertisements to vehicles within vicinity to attract consumers as well as providing dining and parking information.

Although disseminating data from a server to a large number of clients has been studied in the database community and the network community [59, 60], many unique characteristics of VANET bring out new research challenges. First, due to fast vehicle movement, the link topology changes rapidly [22, 23, 61]. As a result, many well studied structures for efficient data dissemination, such as tree [62], clustering [63], grid [64], are extremely hard to set up and maintain. Second, the conventional broadcast mechanism for data dissemination may lead to broadcast storm [24] because the network node density is usually quite high in the urban area, and extremely dense during rush hours or traffic jam. Third, the
vehicle mobility is partially predictable since it is limited by the traffic pattern and the road layout [65]. Data dissemination techniques should address these unique characteristics of VANET.

In this chapter, we propose a data pouring and buffering scheme to address the data dissemination problem in VANET. The proposed solution can reliably disseminate the data and efficiently utilize the limited bandwidth. As a result, it maximize the amount of data that can be disseminated in a given area (i.e., the dissemination capacity). In Data Pouring (DP), data are periodically broadcast to the vehicles on the road. As data are poured along the roads, they are delivered not only to the vehicles on these roads, but also to vehicles on the intersecting roads when they move across the intersections. To further improve the performance, we propose an improved DP scheme called DP with Intersection Buffering (DP-IB), which tries to reduce the amount of data poured from the source by buffering and rebroadcasting data at the intersection.

Periodically pouring data on the road is necessary since vehicles receiving the data may move away quickly and vehicles coming later still need the data. With DP, the data are consistently available for vehicles crossing the dissemination area. In case there are large amount of data from many information sources to disseminate, it is important to increase the amount of data that can be disseminated on a given road. Thus, we provide analytical models to explore the dissemination capacity of the proposed schemes. The analytical models also provide guidelines on choosing the system parameters to maximize the dissemination capacity under different delivery ratio requirements. Extensive simulations are provided and used to evaluate the proposed methodology. Simulation results show that the proposed DP-IB scheme can significantly improve the data delivery ratio and reduce the network traffic.

The rest of this chapter is organized as follows. Section 4.2 describes the data pouring and buffering schemes. Section 4.3 presents an analytical model to optimize the system parameters of our data pouring schemes. Performance evaluations are presented in Section 4.4. Section 4.5 concludes the chapter.
4.2 Data Pouring and Buffering on the Road

In this section, we first describe our system model, and then present the data pouring and buffering schemes.

4.2.1 The System Model

VANET can be used to disseminate traffic accident or traffic jam to help drivers select better routes. It can also be used to disseminate sale advertisements to attract customers. One way of achieving this goal is to have a data center disseminate the data. The data center can be a computer with a wireless interface, a wireless access point, or an infostation [9]. A data center may have a list of data items to disseminate, referred to as the dissemination data set (D-Set). The data center periodically broadcasts the D-Set so that each data item is broadcasted once in each cycle. The disseminated data are relayed by moving vehicles and poured to the desired area.

The data delivery information such as source id, source location, packet generation time, propagation direction and etc, is specified by the data center and placed in the packet header. The disseminated data are often spatial or/and temporal sensitive. For example, the traffic jam at downtown is not likely to be the interest of drivers thirty miles away, and will also be less helpful two hours later. Thus, a data item is attached with two more attributes: the dissemination zone (D-Zone) which is defined as a rectangle area and the expiration time which specifies the duration when the data item is valid. Vehicles outside of this D-Zone will discard the data to save bandwidth.

We assume vehicles communicate with each other and with the data center through short range wireless channel (100m-250m). A vehicle knows its location by triangulation or through GPS device, which is already popular in new cars and will be common in the future. Vehicles use periodic beacon messages to report their moving velocity, direction and location, so each vehicle (including the data center) can get the information about their one-hop neighbors, and construct a neighbor list. To avoid overloading the channel with too many beacons, a vehicle can adjust its beacon interval based on its moving velocity, e.g. two beacons per second when moving over 40 miles/hour, while one beacon every 10 seconds when
moving below 5 miles/hour. Techniques to achieve this can be found in [34, 27].

4.2.2 Data Pouring (DP)

In this section, we first introduce the basic idea of the DP scheme and then propose solutions to make it more reliable.

4.2.2.1 The Basic Idea

The DP scheme makes use of the partially predictable vehicle mobility limited by the road layout. Instead of spreading data throughout the network, it broadcasts the data to one or several roads, called axis roads (A-Roads). The A-Roads are selected from those main roads going through the data center and they normally have higher vehicle traffic density than other roads. The DP scheme also delivers data to vehicles moving on the roads that intersect with the A-Road, called crossing roads (C-Roads). However, it does not proactively push data to the C-Roads. Since vehicles on the C-Roads moving towards the A-Roads will eventually reach a point intersecting with A-Roads, they will get the data. Therefore, the D-Zone of a data item only includes the A-Road where this data item is propagated to. Figure 4.1 shows the basic idea of the DP scheme. Both data centers, P and Q, select the horizontal road A-1 as the A-Road, and the vertical roads C-1 and C-2 as the C-Roads. The data are broadcasted along Road A-1. Figure 4.1 only shows the data broadcasted from the data center P; the circles represent its broadcast coverage, which only covers Road A-1. Vehicles on Roads C-1 and C-2 receive the data when they go through the intersections.

Figure 4.1. Directional Broadcast
In the DP scheme, the data center specifies the road to propagate data based on D-Zone, and adds this information to the packet header. Then it designates a passing by vehicle to broadcast the data (for example, vehicle $a$ in Figure 4.1). To propagate the data to the desired road, the data item needs to be consecutively broadcasted along the road by other vehicles such as $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \rightarrow f \rightarrow g \rightarrow h$ in Figure 4.1. To deal with the broadcast storm problem [24], each designated broadcasting vehicle selects one vehicle that is farther away in the data propagation direction from its neighbor list, and designates the selected neighbor as the next broadcast node by adding it to the packet header. After receiving the data, the designated vehicle (the forwarder) rebroadcasts the data. In this way, the data are poured to vehicles on the A-Roads. A forwarder delivers the data to all potential receivers within its one hop range, and designates one vehicle as the next forwarder to broadcast the data farther along the propagation direction.

Data invalidation [66] in the DP scheme is simple, because vehicles do not cache the data. If the disseminated data are updated, the data center broadcasts the updated version, and vehicles will get the updated data.

4.2.2.2 Reliable DP

The broadcasted data may be lost due to interference, packet collisions and hidden node problem. To ensure vehicles receiving the disseminated data, vehicles use RTS/CTS handshakes to reduce collisions and hidden node problem. To make the broadcast more reliable, after the broadcasting, the sender holds the data packet for a short period of time in case retransmissions are needed. The rebroadcast from the next forwarder is used as an implicit acknowledge to the previous forwarder. If the sender does not hear the rebroadcast from the next forwarder within a pre-specified time period, it selects another neighbor as the next forwarder and rebroadcasts the data.

When the sender is waiting for the rebroadcast by the next forwarder, it may receive some other broadcasted data towards the same direction. In this case, the sender has to buffer the data until the pending acknowledgement has been received from the next forwarder. There are two reasons for blocking the packet forwarding. First, if the selected next forwarder does not broadcast within a given period of time, it may indicate that there is an error on the selected next forwarder; e.g.,
the sender has selected an incorrect next forwarder or the selected forwarder is no longer within the transmission range. This may happen if the neighbor list is outdated or the sender mis-estimate the position of the selected forwarder. As a result, it should choose a different forwarder. Second, even if the next forwarder is correct, it may not be able to rebroadcast when the broadcasting load is too high. Adding more traffic may create congestion and result in more packet loss.

The reliable DP scheme can improve the data delivery ratio even if data are disseminated many hops away from the data center. However, it complicates the transmission with more control messages and backoff procedures, reducing the broadcast throughput. For example, when large amount of data are poured from many data centers independently, the data collision probability is very high. The reliable transmission mechanisms will be frequently used, and lots of bandwidth will be wasted by backoff timers, control messages, and RTS/CTS handshake. This will affect the dissemination capacity.

4.2.2.3 Dissemination Capacity

One desired goal of data dissemination is to maximize the dissemination capacity while ensuring a good data delivery ratio. Simply increasing the D-Set size is not an effective way to increase the dissemination capacity since most data may not be successfully delivered to vehicles if the D-Set size is too large.

Data delivery is subject to two constraints. The first constraint is the data broadcast cycle, i.e. the time interval between broadcasting the same data item on an axial road. To deliver data to moving vehicles, the data are periodically broadcasted. If a vehicle is on the A-Road, it will receive the disseminated data sooner or later. However, vehicles on the C-road can only get the broadcasted data during a short time period; i.e., when they go cross the intersection. During this time period ($t_i$), the vehicle should stay inside the wireless coverage centered at the intersection. Thus, $t_i$ is decided by the wireless coverage and the vehicle moving speed. A vehicle may miss the data if the broadcast cycle is longer than $t_i$. Although reducing the broadcast cycle time can solve this problem, it reduces the dissemination capacity and increases the network traffic. The second constraint is the bandwidth limit. Given a data broadcast cycle time, only limited amount of data can be broadcasted within one cycle. Disseminating data over this limit
will cause collisions and data loss. As a result, the \textit{dissemination capacity (DC)}
within the given D-Zone is equal to the maximum number of data items that can
be broadcasted to the D-Zone in one cycle, which is given by Equation 4.1.

\[ DC = \frac{S \times T}{D_{\text{avg}}} \]  \hspace{1cm} (4.1)

where \( S \) denotes the throughput achievable by multi-hop broadcast in the D-Zone,
\( T \) denotes the broadcast cycle time, and \( D_{\text{avg}} \) denotes the average data size.

Equation 4.1 clearly shows two key factors that limit the dissemination capacity
of the DP scheme. First, the DP scheme reduces the broadcast throughput
since many control messages are used to improve the data delivery ratio. Second,
in a given D-Zone, the location which requires the shortest broadcast cycle time
determines the overall broadcast cycle for the entire dissemination zone. However,
vehicles moving on other part of the road may not need such frequent broadcast,
and vehicles moving along the A-Road indeed need much lower broadcast fre-
quency. Thus, a large amount of bandwidth is wasted in the DP scheme. These
drawbacks motivate our design of the following scheme.

\subsection*{4.2.3 Data Pouring with Intersection Buffering (DP-IB)}

The DP-IB scheme follows the basic idea of the DP scheme, where the data are
poured on the \textit{A-Road}, and vehicles on the C-Roads get data when crossing the
intersections. Instead of keeping the data on the A-Road, DP-IB only keeps the
data at the intersections of the A-Road.

DP-IB relies on a simple device called \textit{relay and broadcast station (IBer)} to
improve the dissemination capacity. IBer can be the popular roadside units [67]
widely used in many VANETs, or a simple computing device with small amount
of memory and a wireless card (e.g. 802.11b). These stand alone IBers can be
easily installed at the intersection. Since they are not required to connect to the
wired network, the deployment cost is low. The IBer is used to buffer data copies
and rebroadcast them periodically. As a result, the data center does not need to
ensure delivering data to the end user. Instead, it only transmits data to those
IBers on the A-Road. In other words, the data center does not have to frequently
broadcast data to guarantee that the vehicles from C-Roads receive the data. The
IBers ensure that vehicles passing the intersection can still get the data, although the frequency required to pour data from the data center is significantly reduced. Further, the IBers can adapt different broadcast cycle time at different intersection, i.e., longer cycle for slow moving traffic, while shorter cycle for fast moving traffic. Two issues still need further investigation:

1. How to reliably upload the data from the data center to the IBers and then deliver to the end users?

2. How to choose different broadcast cycle time for different parts of the road?

Next, we address the first problem and leave the second to Section 3.2.

4.2.3.1 Intersection data buffering and rebroadcasting

The data center in DP-IB pours data on the A-Road using the reliable DP scheme. When data are forwarded through the intersections, the IBers overhear the data and update their own buffers accordingly, i.e., insert new data item, update existing data to a new version, or remove the invalid data. The IBer will broadcast the updated data in the next broadcast cycle. Since the IBer uses a single hop broadcast to deliver data to vehicles, the data broadcast throughput is high.

One problem may arise when a new data item from the data center is transmitted to the intersection, while the IBer is in the middle of broadcasting its buffered data. It is possible that the transmission of the new data item collides with the IBer broadcast, and the IBer cannot receive the new data. Our solution is to have the IBer and the forwarding nodes alternatively obtain the channel for broadcast. More specifically, the IBer broadcast cycle is divided into two periods: in the first period (called busy period), the IBer broadcasts its buffered data while the forwarding nodes temporarily hold their data. In the second period (the idle period), the forwarding nodes forward pending packets, while the IBer stops broadcasting and only listens to the channel for passing-by data packets. In this way, the IBer releases the channel for a period of time in every broadcast cycle so that new data items can be received.

Intersection contention avoidance protocol: All vehicles switch between two modes: active and inactive forwarding mode. They stay in the active forwarding
mode most of time, and only switch to inactive when they are inside an IBer broadcast range and the IBer is in the busy period at the same time. When the vehicle is in the active forwarding mode, it forwards data using the reliable DP scheme. It switches to inactive immediately after receiving a broadcast data packet from an IBer. In the inactive forwarding mode, the vehicle stops forwarding data until it goes back to active.

The IBer broadcasts an \textit{IBer\_Idle} message when its busy period ends. All vehicles receiving this message switch to active forwarding and start to forward any pending data. A vehicle in the inactive forwarding mode may fail to receive the \textit{IBer\_Idle} message due to message loss, or it may move out of the IBer broadcast range when the busy period ends. Thus, the inactive forwarding mode is set as a soft state, and the vehicle switches back to active if it does not receive any broadcast packet from the IBer for a time period.

**Determining the busy/idle period:** If an IBer does not buffer too much data, it can simply send an \textit{IBer\_Idle} message after broadcasting all its buffered data. However, if the amount of data buffered exceeds the data dissemination capacity, the IBer should stop broadcasting before the end of the cycle, and leave some time for the idle period. Equation 4.2 is used to determine the length of the busy period.

\[
L_{est} = \frac{n \times D_{avg}}{B_{IBer}} + \sum_{i=1}^{m} \frac{T_i}{B_{R-DP}} \times D_{avg} 
\]  

(4.2)

In Equation 4.2, \(n\) is the number of buffered data items, \(D_{avg}\) is the average data size, \(m\) is the number of data centers which have stored data at the IBer, and the IBer receives the data packet from the \(i\)th data center every \(T_i\) time interval; \(B_{IBer}\) and \(B_{R-DP}\) represent the measured broadcast throughput of the IBer and the Reliable DP scheme respectively; \(T\) is the actual broadcast cycle of the IBer. The IBer can obtain all the above information locally. \(L_{est}\) estimates the time needed to finish broadcasting all the buffered data, and the time to overhear all the new data packet generated in a broadcast cycle. If \(L_{est}\) is smaller than \(T\), the IBer can finish broadcasting all the buffered data, and then send the \textit{IBer\_Idle} message. The rest of the broadcast cycle still allows the IBer to overhear all the new data. When \(L_{est}\) is larger than \(T\), the IBer is only given \(\alpha T\) in a cycle to broadcast data before sending out the \textit{IBer\_Idle} message, where \(\alpha\) is a system parameter, and it
is much smaller than 1. In the rest of the broadcast cycle \( (1 - \alpha)T \), the IBer enters the idle period.

### 4.2.3.2 Data Update and Invalidation

When new data items are added or old data items are updated at the data center, the data center immediately broadcasts the new data or the invalidation message using the reliable DP scheme. The data center may occasionally rebroadcast the data that have been disseminated, in case the IBer fails to obtain the data. Since all copies of the disseminated data are only buffered at the intersections along the A-Road, one invalidation message sent along the A-Road will be able to remove the invalid copies.

### 4.2.3.3 Data Dissemination Capacity

The dissemination capacity of DP-IB is limited by the broadcast cycles of the IBers. Compared with the DP scheme, the number of control messages in DP-IB is significantly reduced, and hence can speed up the data transmission. Although an idle period is added to the broadcast cycle, it only takes a small portion of the broadcast cycle time, while the majority of the broadcast cycle time is spent on data broadcast. Thus, it has better bandwidth utilization than DP, where each data packet transmission may involve many control messages and extra backoff times. As a result, the throughput of data broadcast in DP-IB is much higher than DP, and the dissemination capacity is higher.

### 4.3 Analyzing and Determining The Broadcast Cycle Time

In this section, we use an analytical model to determine the broadcast cycle time at the intersection, denoted as \( T_i \). \( T_i \) is used to determine the dissemination capacity and the delivery ratio of the DP scheme and the DP-IB scheme. \( T_i \) is also closely related to the time for a vehicle going through the intersection region, denoted as \( t_i \). Intuitively, \( T_i \) should be less than the minimum \( t_i \) of all vehicles moving through the intersection region to guarantee that all vehicles passing the intersection can receive
the broadcasted data. If disseminating more data becomes the main objective, we can improve the dissemination capacity by increasing $T_i$. This is at the cost of reducing the data delivery ratio, since some vehicles moving across the intersection may miss part of the data. To derive $T_i$, we first model the intersection delay.

### 4.3.1 Modeling the Intersection Delay

Although there are many different intersection structures in reality, such as signalized, isolated, roundabout and etc., our intersection delay model only studies the vehicle delay at the signalized intersection with two crossing paths, because it can simplify the presentation and still show the relation between the network properties and the vehicle traffic properties. Our analysis can be easily extended to more complicated intersections, where the vehicle delay distribution is usually modeled by applying advanced transportation traffic theory or empirical traffic flow statistics, which is beyond the scope of this dissertation.

The delay $t_i$ can be represented by two parts\cite{68}: the first part, referred to as the moving delay (denoted as $t_m$), is the ideal travel time if this vehicle does not meet any traffic control signal. It is equal to the moving distance at the intersection divided by the normal moving speed of the vehicle. The second part, referred to as the queuing delay (denoted as $t_q$), is the extra delay caused by the traffic signal, including deceleration delay, stopped delay and acceleration delay. Their relations are given by:

$$t_i = t_m + t_q.$$  \hspace{1cm} (4.3)

#### 4.3.1.1 Calculating $t_m$, $t_q$ and $t_i$

$t_m$ is related to the vehicle moving distance and speed on the C-Road covered by the data broadcast. The broadcast node is always on the A-Road, and the broadcast can reach a fixed portion of the C-Road. Given the length of the C-Road covered by the broadcast as $R$, $t_m$ can be easily computed as

$$t_m = R/v$$ \hspace{1cm} (4.4)

where $v$ is the normal speed of the vehicle moving across the intersection.
To get the minimum \( t_i \), vehicles move through the intersection at its normal speed without deceleration and stop. Thus, \( t_i^{\text{min}} = t_m \). In reality vehicles rarely keep the normal speed at the intersection because of traffic control signals. Most vehicles experience deceleration, acceleration, and often wait in line with full stop [69]. Not all the vehicles experience the same delay when traveling through an intersection. The delay depends on a number of factors such as the traffic flow density, signal time and the time when the vehicle arrives at the intersection. Based on an example about 11 vehicles shown in [69], the first eight vehicles reaching the intersection come to a complete stop. These vehicles need to stop either as a consequence of their arrival during the red interval or during the green interval when the queue of vehicles that had formed during the previous red interval has not yet fully dissipated. It is further observed that the following three vehicles only experience deceleration and acceleration delay, as these vehicles reach the intersection when all previously queued vehicles have already started to move and therefore only need to slow down to maintain a safe distance with the vehicles ahead of them.

![Figure 4.2. Analytical model for \( t_q \)](image)

We study the intersection delay with under-saturated traffic flow, which means the arrival rate \( r_a \) is less than the saturated departure rate \( r_d \), and we assume that the queue of vehicles formed during the red signal cycle can always be cleared before the next red signal.

Let \( t_r \) and \( t_g \) denote the red and green signal duration respectively. Suppose a vehicle arrives at the intersection \( t \) seconds after the red signal turns on, as shown in Figure 4.2, \( t_q \) of this vehicle is given by:

\[
t_q = \begin{cases} 
  t_r - t + \frac{r_a}{r_d}, & t \in \left[0, \frac{r_a t_r}{r_d - r_a}\right] \\
  0, & \text{else}
\end{cases} \tag{4.5}
\]
In Equation 4.5, \( \frac{r_d t_r}{r_d - r_a} \) shows the time it takes to clear the queue accumulated at the intersection after the red signal starts. When the vehicle arrives at the intersection after that time, it will go through the intersection without delay; if the vehicle arrives before that time, it waits for the vehicles that queued at the intersection to depart, and the delay can be calculated by Equation 4.5.

We assume the vehicle arrival follows uniform distribution, which is widely used in traffic flow modeling [69]. The probability distribution function (PDF) of \( t_q \) is given by:

\[
f_{t_q}(x) = \text{Prob}\{t_q = x\} = \begin{cases} 
0, & x < 0 \\
1 - \frac{r_d t_r}{(r_d - r_a)(t_r + t_g)}, & x = 0 \\
\frac{r_d}{(r_d - r_a)(t_r + t_g)}, & 0 < x \leq t_r \\
0, & x > t_r
\end{cases} \tag{4.6}
\]

Since \( t_m \) can be computed as a constant from Equation 4.4, the PDF of \( t_i \) (\( f_{t_i}(t) \)) can be easily computed by combining Equation 4.3, 4.4 and 4.6:

\[
f_{t_i}(x) = \text{Prob}\{t_i = x\} = \text{Prob}\{t_q = x - t_m\} = f_{t_q}(x - t_m) \tag{4.7}
\]

### 4.3.2 Determining the Broadcast Cycle Time

Only when the data broadcast cycle time \( T_i \) at the intersection is smaller than the minimum vehicle delay \( t_i \), it is possible to deliver all data in D-Set to every moving vehicles on the C-Road. We define this broadcast cycle time which allows all vehicles to receive all data in D-Set as full delivery broadcast cycle time, denoted by \( \overline{T_i} \), and it is given by:

\[
\overline{T_i} \leq t_m = \frac{R}{v} \tag{4.8}
\]

Suppose the transmission range of the broadcast is 100 meters, and the C-Road perpendicularly intersects with the A-Road. Then, the length of the C-Road covered by the broadcast \( R = 200m \). Assume the normal vehicle speed at the intersection is 45 miles/hour (20.1 meter/sec). Then, \( \overline{T_i} \) is less than 9.95sec, i.e. the broad-
cast node at the intersection has to finish broadcasting all data in the D-Set in 9.95 seconds.

In Simple DP and Reliable DP, all intersections on the A-Road and the data center use the same broadcast cycle time, which is computed by

\[ T_{DC} = \min_{i \in A-Road} \{T_i\}. \]  \hspace{1cm} (4.9)

Thus, the intersection which requires minimum broadcast cycle time among all the intersections determines the broadcast cycle time. However, in DP-IB, different intersections use different broadcast cycle time. It is controlled by the local broadcast node IBer, and can be adaptively adjusted based on the local vehicle traffic condition.

4.3.3 Relations between Delivery Ratio, Broadcast Cycle Time and Dissemination Capacity

Equation 4.1 shows that the dissemination capacity is linear to the broadcast cycle time. From the previous section, \( T_i \) sets the upper bound for the broadcast cycle time to ensure the best delivery ratio, which also bounds the maximum dissemination capacity. By sacrificing some data delivery ratio, we can increase \( T_i \) and broadcast more data. Next, we study the effects of \( T_i \) on the data delivery ratio, and explore the trade-off between dissemination capacity and data delivery ratio.

Given that the PDF of \( t_i \), \( f_{t_i}(x) \), can be computed by Equation 4.7, the data delivery ratio at the intersection (denoted as \( DR \)) can be computed as:

\[ DR = \int_{T_i}^{\infty} f_{t_i}(x) \, dy + \int_{0}^{T_i} f_{t_i}(x) \cdot \frac{x}{T_i} \, dx \]  \hspace{1cm} (4.10)

Equation 4.10 shows that vehicles staying longer than \( T_i \) at the intersection (i.e., \( t_i > T_i \)) can receive all the broadcasted data, while those staying shorter than \( T_i \) (i.e., \( t_i < T_i \)) only receive \( \frac{t_i}{T_i} \) portion of the broadcasted data.

By combining Equation 4.6, 4.7 and 4.10, the relation between \( DR \) and \( T_i \) can
be derived as:

\[
DR = \begin{cases} 
1, & T_i \leq t_m \\
\frac{t_m \left((r_d-r_a)(t_r+t_g)-r_d t_m - \frac{1}{2} r_d t_m^2\right)}{(r_d-r_a)(t_r+t_g) T_i} & t_m < T_i \leq t_m + t_r \\
\frac{r_d(t_m+t_g)-\frac{1}{2} r_d T_i}{(r_d-r_a)(t_r+t_g)}, & T_i > t_m + t_r \\
\frac{2t_m(t_r+t_g)(r_d-r_a)+r_d t_m^2}{2T_i(r_d-r_a)(t_r+t_g)}, & T_i > t_m + t_r 
\end{cases}
\]

(4.11)

In the above equation, the signal length \( t_g \) and \( t_r \) can be seen as fixed road properties. Parameter \( r_d \) represents the capability of the intersection to disperse vehicle traffic, which is usually fixed after the road has been constructed. Parameter \( r_a \) indicates the dynamic vehicle traffic load at the intersection. Therefore, with \( t_g = 30s, t_r = 30s, t_m = 10s, r_d = 30 \), the relations among \( T_i, DR \) and the local vehicle traffic load are shown in Figure 4.3.

![Figure 4.3](image)

**Figure 4.3.** Analyze the relation between the broadcast cycle time and the data delivery ratio.

Figure 4.3 clearly shows that the dissemination capacity and the data delivery ratio are conflict design goals. If we are given the delivery ratio requirement, for example \( DR > 90\% \), we can compute the largest \( T_i \) based on Equation 4.11. This \( T_i \) value provides the maximum dissemination capacity while keeps the delivery
ratio equal to 90%. Using the above relations, we can tradeoff these two metrics for the best broadcast strategy based on the requirement of the applications. This figure also addresses the second question raised in Sec 4.2.3.1. The figure shows the relations among the broadcast cycle time, the data delivery ratio and the vehicle traffic load. Generally speaking, when $T_i$ increase, the data delivery ratio drops. As the vehicle traffic density increases, vehicles stay longer at the intersection because they move slower and stop longer. Thus, more data can be delivered to drive-through vehicles by extending the broadcast cycle time at the intersection without reducing the data delivery ratio. Also, if the passing through vehicles only receive part of the D-Set, without changing the broadcast cycle time, the dissemination ratio will be higher as the vehicle traffic density increases. This provides guidelines for DP-IB to dynamically adapt its broadcast cycle time based on the local vehicle traffic load at different intersections.

4.4 Performance Evaluations

In this section, we evaluate the performance of the Simple DP scheme (without using techniques presented in Section 4.2.2.2), the reliable DP scheme, and the DP-IB scheme. We also compare these schemes to the opportunistic dissemination (OD) scheme presented in [44], and the MAC-layer based reliable multi-hop broadcast scheme (UMB) proposed in [40].

4.4.1 The Simulation Model

We developed an ns-2 [70] based simulator to evaluate the proposed schemes. The simulation is based on a $4500m \times 600m$ rectangle area extracted from a real street map of State College, Pennsylvania. The position of the data center and the D-Zone are shown in Figure 4.4, represented by the star and the crossing rectangle respectively. We choose East College Avenue, one of the most crowded streets in State College, as the A-Road. It runs through the downtown with 25 miles/hour speed limit. The C-Roads are selected from three major streets that intersect with the East College Avenue. All the three C-Roads have the same speed limit (45 miles/hour). The street layout and speed limit information are translated
into a text format map which meets the specification of Topologically Integrated Geographic Encoding and Referencing (TIGER) database [57] from U.S. Census Bureau. These text-based map data are then transformed into the data format that can be used by ns-2, based on techniques presented in [58]. In order to simulate the vehicle traffic on the A-Road, we initially randomly deploy 150 vehicles on the A-Road and let them move towards either end of the road. Those vehicles move back and forth with 25 miles/hour during the simulation to mimic continuous traffic flow along the A-Road. We assume the vehicle density on the A-Road is large enough to maintain network connection, which is possible in urban areas where our data dissemination scheme is designed for.

Since vehicles move along the A-Road can always receive the disseminated data, we are more interested in vehicles on C-Roads. Among vehicles on the C-Roads, only those close to the intersections are relevant to data dissemination. Since simulating the movement of vehicles across the intersection is more important than the traffic beyond the intersection, we only consider the vehicle traffic on a
600-meter long section of C-Roads, whose middle point intersects with the A-Road. We initially deploy 20 nodes at the upper end of each C-Road, and let them move back and forth between the two ends of the C-Roads. When the node arrives at the intersection, it stops for a random amount of time with the distribution given by Equation 4.6 before moving again. When the node arrives at the other end of the road section, it pauses for a time period, and moves back. Each vehicle on C-Roads randomly picks a value between 15 to 45 (miles/hour) as its moving speed. Figure 4.5 shows a snapshot of the simulation area. The data center is located at the left end of the A-Road, and disseminates data along the A-Road towards the right.

Simple DP and Reliable DP use the same broadcast cycle time throughout the A-Road. DP-IB can adapt the IBer broadcast cycle time based on the local vehicle traffic at the intersection. In our simulation model, the same traffic pattern is used in all the C-Roads, and hence we use the same intersection broadcast cycle time for all intersections.

At the data center, all data in the D-Set are repeatedly injected to the A-Road. The data items are sent by the data center one after another with a given time interval. In Simple DP and Reliable DP, this time interval is equal to the broadcast cycle time divided by the number of data items to be disseminated. In DP-IB, since the frequency required to pour data from the data center can be greatly reduced, it takes much longer time to broadcast the whole data set at the data center. Thus, the time interval is set as 250 ms. We only consider data items with fixed size of 2500 bits. Each vehicle sends a beacon message every 0.5 second to report its own location and speed.

Most experiment parameters are listed in Table 6.4. The performance of the protocols is measured by the following two metrics:

- **Data delivery ratio**: For each vehicle, the data delivery ratio is the total number of nonidentical data items received divided by the total number of data items disseminated.

- **Network traffic overhead**: The number of bits generated per second, which is a summation of individual packet-hops. For instance, if a packet of 1000 bits is forwarded 10 hops, the network traffic overhead is counted as 10K
Table 4.1. Simulation Setup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation time</td>
<td>500 seconds</td>
</tr>
<tr>
<td>Simulation area</td>
<td>$4500m \times 600m$</td>
</tr>
<tr>
<td>Communication range</td>
<td>100m</td>
</tr>
<tr>
<td>Number of nodes on the A-Road</td>
<td>150</td>
</tr>
<tr>
<td>Number of nodes on the C-Roads</td>
<td>60</td>
</tr>
<tr>
<td>Vehicle velocity</td>
<td>15 - 45 miles per hour</td>
</tr>
<tr>
<td>Data packet size</td>
<td>2500 bit</td>
</tr>
<tr>
<td>Dissemination data set size</td>
<td>10 - 600 data items</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>10Mbps</td>
</tr>
<tr>
<td>Intersection broadcast cycle time</td>
<td>5 - 40 s</td>
</tr>
<tr>
<td>Beacon interval</td>
<td>2 beacon/second</td>
</tr>
<tr>
<td>Beacon packet size</td>
<td>512 bit</td>
</tr>
<tr>
<td>Control packet size</td>
<td>512 bit</td>
</tr>
<tr>
<td>DP-IB cache replacement policy</td>
<td>FIFO</td>
</tr>
</tbody>
</table>

For each measurement, 30 simulation runs are used and a different seed value is used for each simulation run. For the data delivery ratio, the mean value of the measured data is obtained by collecting a large number of samples such that the confidence interval is reasonably small. In most cases, the 95 percent confidence interval for the measured data is less than 10 percent of the sample mean.

Figure 4.6. Comparison of the data delivery ratio (with the broadcast cycle of 10s)
4.4.2 Simulation Results

4.4.2.1 The relation between delivery ratio and dissemination capacity

Figure 4.6 shows the relation between delivery ratio and the amount of data to be disseminated (dissemination capacity) for the five schemes. When only a small amount of data items are disseminated (e.g., 10 data items in Figure 4.6), there are plenty of bandwidth available and the data delivery ratio of all schemes are close to 100%. Among them, the OD scheme has slightly higher delivery ratio than others, because OD explores every possible path to propagate the data. However, the delivery ratio of OD becomes much lower than DP and DP-IB when more data items (above 50) are disseminated. As discussed in the introduction, the OD scheme generates too much redundant network traffic, which may lead to severe congestion and significantly reduce the data delivery ratio. When the D-Set size is above 50, there is a significant increase of data losses in the Simple DP scheme, and its data delivery ratio drops dramatically. Therefore, the dissemination capacity of the Simple DP scheme will be very low to maintain a good data delivery ratio.

UMB, Reliable DP and DP-IB have very high data delivery ratio when the data set size is below 150. When more data are to be disseminated, the delivery ratio of the UMB scheme and the reliable DP scheme drops quickly, while the DP-IB scheme keeps the same data delivery ratio. The data delivery ratio of UMB drops sharply because the packets in a dissemination data flow interfere with each other. As the channel usage increases, the interference problem becomes worse, leading to severe packet loss. Although UMB can effectively improve the reliability of multi-hop broadcast, this MAC layer approach cannot address network congestion. The low delivery ratio of Reliable DP is not because of packet loss, since the packet can still be reliably transmitted. Thus, the delivery ratio of Reliable DP drops gracefully compared to UMB. However, the reliable DP scheme reaches the saturate status when the D-Set size reaches 150. When more data need to be disseminated, reliable DP cannot finish broadcasting the whole data set within one cycle. As a result, some vehicles move across the intersection, without receiving all the data. DP-IB can broadcast data with higher throughput, and will not reach the saturate state until 360 data items are broadcasted. Thus, it has more dissemination capacity than the reliable DP scheme. When the D-Set size is more
than 360, the IBer in the DP-IB scheme may not be able to finish broadcasting the whole data set within its broadcast cycle time, and the data delivery ratio decreases.

4.4.2.2 Revisiting the relations between delivery ratio, broadcast cycle time and dissemination capacity

Figure 4.6 only shows the results under a fixed broadcast cycle time (10 seconds). To further examine the relations among the broadcast cycle $T_i$, the data delivery ratio and the dissemination capacity, we present more results in Figure 4.7 when the D-Set size and the broadcast cycle time change. Figure 4.7 shows additional relations not shown in Figure 4.3 which does not consider the network bandwidth limit. In Figure 4.3, we assume the whole D-Set can be transmitted. If vehicles fail to receive the data, it is because the broadcast cycle time is too long and the vehicles passes the intersection before one cycle finishes (short intersection stay). However, data may not be delivered because the D-Set set is too large. That is, the network is saturated and cannot accommodate all the disseminated data.

Figure 4.7 studies the packet loss caused by both factors: short intersection stay and network saturation, under the same vehicle traffic load. In Figure 4.7, the 3D surface in each figure can be divided into four regions, based on the factors causing the packet loss. As shown in Figure 4.7(a), in Region 1, there is no packet loss because of the above factors, and the delivery ratio is close to 100%. In this region, the broadcast cycle time is smaller than 10 seconds. Then, there is no packet loss due to short intersection stay since even the fastest vehicle takes 10 seconds to move across the intersection. Also, the D-Set size is small, and the network is not saturated. In Region 2, the only factor for packet loss is network saturation, since the D-Set cannot be delivered within 10 seconds. There is no packet loss due to short intersection stay as the broadcast cycle time is still less than 10 seconds. In Region 3, the packet loss is due to short intersection stay, since the broadcast cycle time is longer than 10 seconds, and some vehicles may pass the intersection without getting the data. There is no packet loss due to network saturation, because the D-Set can be sent within the given broadcast cycle time. Thus, within this region, the D-Set size can be increased without affecting the data delivery ratio. In Region 4, the D-Set size exceeds the network capacity and the
broadcast cycle time is longer than 10 seconds, so packets are lost for both factors.

Figure 4.7. Relations among cycle time, data set size and delivery ratio

Figure 4.7 (b) and (c) can also be divided to four regions using the same criteria, but the areas of the same region are different in these three schemes. The areas of Region 1 and Region 3 in DP-IB are much larger and flatter than the other schemes. It means that DP-IB can always disseminate the largest amount of data without reaching network saturation. In Simple DP, there is no mechanism to reduce the collision, so the bandwidth utilization is very low. Therefore, the areas of its Region 2 and Region 4 are very large while the areas of its Region 1 and Region 3 are quite small. In most cases, the network is saturated and less data can be successfully transmitted without collision or being dropped.

Overall, given any required data delivery ratio, DP-IB can always disseminate
the largest amount of data when a proper broadcast cycle time is chosen. Reliable DP can disseminate less data than DP-IB, but still far more than the Simple DP scheme under any data delivery ratio requirement.

4.4.2.3 The effects of bit error rate

![Graph](image.png)

**Figure 4.8.** Impact of the transmission bit error rate

Figure 4.8 compares the four schemes with different transmission bit error rate when 50 data items are disseminated. The simple DP scheme is based on simple broadcast, so the data delivery ratio quickly drops when the error rate increases. The data delivery ratio of the reliable DP scheme only drops slightly when the error rate increases, which verifies the effectiveness of the reliable multi-hop broadcast. The OD scheme can also keep a good delivery ratio due to its opportunistic nature. Among the four, DP-IB is the most resilient to bit errors. Since the data are buffered at the intersection, the impact of bit error on multi-hop transmission is minimized.

4.4.2.4 Dissemination distance

Figure 4.9 shows the data delivery ratio of the three DP schemes when the data are delivered to vehicles far away from the data center. The D-Set size is 100. As can be seen, the delivery ratios of both Reliable DP and DP-IB are not affected by the dissemination distance. This also verifies the effectiveness of the reliable multi-hop broadcast scheme in handling transmission errors. On the other hand, the delivery ratio of the Simple DP scheme drops quickly as the distance increases,
because packet loss occurs at every hop and not much data left after a long-distance pouring.

4.4.2.5 Dissemination delay

Figure 4.10 shows the delay of delivering a new data item to vehicles at three intersections using the UMB scheme and the three DP schemes. The D-Set size is set to 50. The delay is computed from a new version of a data item generated at the data center to the time when the first vehicle receives it at the specific intersection. As shown in the figure, UMB has the lowest delay since it does not block the data flow, and does not use any timers. The simple DP scheme does not use any reliable transmission mechanism, but the delay is still longer than UMB, because some new data items fails to reach the intersection in the first cycle after being generated. When it is delivered in the second cycle, the delay is increased by one cycle time (10 seconds). The reliable DP scheme has higher delay since it blocks the flow until the ACK is received for the previous packet, and this increases the packet forwarding latency. DP-IB has the highest delay due to the following reason. After a new data item has been generated, it needs to be uploaded to the IBer, and the upload only starts after the IBer finishes its broadcast cycle. Hence, the new data item needs to wait for half of the broadcast cycle time on average before transmitting the data to the IBer. However, the delay is at the level of seconds and only happen when the accessed data item is updated. Most of time, this delay does not exist. Thus, this dissemination delay should not be a big issue compared to other factors such as dissemination capacity and data delivery ratio.
4.4.2.6 Network traffic overhead

Figure 4.11 compares the overall network traffic overhead generated by the five schemes when the D-Set size increases. The simple DP scheme generates the least amount of traffic, since it simply broadcasts the data, without adding any control message. Further, many data packets are dropped before going too far on the A-Road. This can be reflected by the low data delivery ratio shown in Figure 4.6.

The OD scheme has the highest traffic overhead since vehicles keep sending data to each other. The traffic overhead of the reliable DP scheme is also pretty high, because many control messages are generated for each data packet transmission and it introduces many retransmissions to improve the reliability. DP-IB generates much less traffic compared to Reliable DP, since DP-IB delivers data to end nodes by single hop broadcast from the IBers, instead of reliable multi-hop broadcast from the data center. As a result, the data center broadcasts data at a much lower rate, which greatly reduces the network traffic overhead.

These five schemes generate more network traffic when the D-Set size increases at the beginning. After the channel saturates, the disseminated data may be dropped and the traffic only slightly grows or even keeps constant when the number of the dissemination data increases. Figure 4.11 shows that the network traffic of the Reliable DP scheme keeps almost constant and DP-IB only slightly grows when they disseminate more data than their dissemination capacity, which is 150 and 360 data items respectively. Figure 4.11 also shows that when the channel saturates, DP-IB disseminates more data than Reliable DP but still generates less traffic,
which verifies that DP-IB can improve the bandwidth utilization.

Figure 4.11. Comparison of the network traffic

Figure 4.12. Detailed comparison of the network traffic

Figure 4.12 shows the distribution of the network traffic. The control packet includes the beacon packets used by all schemes except UMB, RTS/CTS handshake used by reliable DP and DP-IB, and the control packets for the intersection contention avoidance used by DP-IB. In the figure, all schemes generate the same amount of beacon messages except UMB. As can be seen, all schemes have very small amount of control packets when the network traffic load is low (e.g. to disseminate 10 data items). When the network traffic load increases (e.g. to disseminate 100 data items), the portion of control packets becomes significant for the reliable DP (R-DP) scheme. On the other hand, the control traffic of the DP-IB
scheme is not very high, because the data center can slow down the rate of pushing data to the intersections. For the OD scheme, although it does not have control traffic overhead, its data packet overhead is much higher. As explained earlier, the simple DP scheme drops many packets, and hence the traffic overhead is lower than others.

4.5 Conclusions

In this chapter, we proposed a data pouring and buffering scheme to address the data dissemination problem in VANET. In Data Pouring (DP), data are periodically broadcasted to vehicles on the road. In DP-IB, data poured from the data center are buffered and rebroadcasted at the intersections. Simulation results show that the proposed DP-IB scheme can significantly improve the data delivery ratio and reduce the network traffic. Further, we provide analytical models to explore the dissemination capacity of the proposed schemes. The analytical models also provide guidelines on choosing the system parameters to maximize the dissemination capacity under different data delivery ratio requirements.

The DP-IB scheme is proposed as a general solution for maximizing the dissemination capacity, where different broadcast cycle time is used at the data center and different intersections. As stated in our system model, the DP scheme is designed for environments where the vehicle density is reasonably high to maintain connection. Since the DP scheme is only used to deliver the data from the data center to the intersection, it can be replaced by other schemes without significantly affecting the dissemination capacity. Thus, in sparse environment, store and forward solutions discussed in Chapter 3 can be added to the DP scheme to deal with disconnections.
Chapter 5

Extending The Coverage of Roadside Units

5.1 Introduction

In VANETs, stationary roadside units (RSUs) may help with a wide variety of services. For example, they can be deployed every few miles along the highway for users to download maps, traffic data, and multimedia files [71]. Vehicles can use RSUs to report real time traffic information and to assist other vehicles in avoiding traffic congestion. Although 3G networks or satellite techniques can be used to achieve this goal, RSUs have the advantage of low cost, easy deployment, and high bandwidth.

A normal RSU consists of a radio transceiver (such as 802.11 access point) that provides high bit rate, low cost, low power network connections to vehicles in a restricted coverage area, along with computer hardware and software that store relevant data and schedule transmissions.

To minimize cost, RSUs may be deployed several miles apart. In some cases, the RSUs being accessed are residential RSUs which are not an engineered network [33]. This should not present a problem for connecting to the Internet, as long as vehicles can frequently move through RSU coverage areas. However, unlike stationary users, vehicles move very quickly and only stay within a RSU’s coverage for a short time. Thus, one important challenge is to maximize connection time.
and the amount of data transferred for drive-thru vehicles.

The connection time or the RSU coverage is not enough for measuring the performance of data access. During a given time period, a high-bandwidth connection can be used to transfer more data than a low-bandwidth connection. When a vehicle drives through a RSU’s coverage area, it experiences poor signal strength when entering and leaving the wireless access range, but has strong signal strength near the RSU [32, 72, 31]. A vehicle can establish connectivity with a RSU from a considerable distance, e.g. 100m, but the packet loss rate and the transmission delay are very high at the fringes of the RSU’s coverage area. A vehicle can achieve a stable connection with acceptable throughput within 60m of the RSU. Thus, for vehicles away from 60m, the connectivity to the RSU is not productive.

In this chapter we propose a relay-based solution to extend the service range of roadside RSUs. As a vehicle moves towards a RSU, its signal quality with the RSU may be poor. In order to extend its connection time, the vehicle selects a vehicle geographically ahead of it to serve as a relay. The vehicle also selects a vehicle behind it to serve as a relay when it leaves the RSU coverage area. The relay approach can improve the throughput and extend the RSU coverage. When nodes close to the RSU also need to access the RSU, they may compete bandwidth with nodes asking for relay. However, this kind of problem may be worse without relay. This is because vehicles at the fringe of the RSU coverage area compete with vehicles closer to the RSU no matter there is a relay vehicle or not. Further, IEEE 802.11 is known to suffer from so-called performance anomaly [73, 74]. If the vehicles at fringe of the RSU’s coverage area directly access the RSU using the lossy links, the RSU is forced to adapt to a lower transmission rate to maintain good link quality, reducing the system throughput. The relay vehicle keeps the faraway vehicles from accessing the RSU using one-hop poor link, which can help mitigate the performance anomaly problem for drive-thru vehicles.

Relay via multi-hop ad-hoc networks has already been extensively studied [75, 76, 77, 78]. The work in UCAN [75] focuses on finding a routing path from a mobile device to other mobile devices with better cellular network coverage. The work of rDCF [78] focuses on using a MAC layer based approach to relay, which is more efficient than establishing a relay path at the network layer. However, all these previous schemes cannot be easily applied to drive-thru vehicles because vehicle
mobility can cause frequent interruptions of the relay link, which can significantly degrade the throughput. To the best of our knowledge, this is the first work to study wireless relay for vehicle drive-thru data access from both theoretical and empirical perspectives. We study the car following characteristics in the traffic and design a stochastic model to find reliable links among vehicles. We also develop a viable prototype to identify and resolve the implementation issues. Through simulation, we demonstrate that our solution well handles high vehicle mobility, and allows drive-thru vehicles to maintain high throughput in the extended RSU coverage area.

The reset of the chapter is as follows. Section II gives the motivation of the work. Section III presents our relay scheme. In Section IV, we present the experimental implementations. Section V evaluates the performance of the proposed relay scheme, and Section VI concludes the chapter.

5.2 Motivations

In order to verify our hypothesis that one can use ad-hoc relay to increase the RSU access time, we conducted a simple experiment. Figure 5.1(a) and (b) show the experiment setup. Our testbed consisted of a Linksys wrt54GL wireless router as the RSU, a Linux server, a PowerBook G4 laptop as relay and a Linux laptop as the client. The RSU was mounted on the roof of a two-story house on the roadside in a residential area. The client laptop communicated with either the relay laptop or the RSU using a Prism chipset based Orinoco 802.11b wireless card. The relay laptop communicated with both the client laptop and the RSU using an Airport 802.11a/b/g wireless interface and a Netgear USB card. The server was connected to the RSU through a high speed cable. In order to test the two hop RSU access performance, we configured the relay laptop to forward packets from the client laptop destined to the server by masquerading with IP tables. We also manually changed the routing table of the client laptop, adding the relay laptop as the default gateway so that the client laptop sent all generated traffic to the relay laptop. We incrementally moved the client away from the RSU location, keeping the relay laptop equidistant between the RSU and the client.

We performed two experiments. The first experiment tested what percentage
of the RSU beacons could be received at different distances from the RSU. The second experiment tested the UDP throughput between the client and the server using Netperf [79].

Figure 5.2(a) shows the results of our first experiment. As can be seen, the beacons could be reliably received (above 90%) when the client stayed within 60 meters of the RSU. Outside this range, the delivery ratio had an abrupt drop, and the link became lossy. The results also showed that even though the client was fairly far away from the RSU (e.g. 100 meters), it could still receive 22% of the beacons.

Our second experiment (Figure 5.2(b)) compared one-hop and two-hop UDP throughput at different distances. The points in the figure are the throughput values we collected. We also generated a line of best fit based on the collected data by the regression method. In the figure, the one-hop UDP throughput was consistently above 2.8 Mbps within the 60-meter range from the RSU. Outside this range, an abrupt throughput drop could be seen, and the throughput quickly dropped below 0.5 Mbps. This rapid throughput drop beyond a certain distance was reported in other works [32, 72, 31]. By using the two-hop approach, the throughput decrease was much smaller compared to the single hop approach. As shown in the figure, the two-hop approach could still achieve good throughput (close to 2 Mbps) outside the 60-meter range of the RSU, while the single hop approach was below 0.5 Mbps. It also extended the RSU access range with fairly high and stable throughput (above 1.8 Mbps) to 100 meters, almost twice as far as where stable throughput could be achieved by the one-hop approach. However,
the throughput of the one-hop approach was better than that two-hop approach within the 60-meter range. We also find that the location where the abrupt drop of the one-hop throughput coincided with the location where the massive beacon loss was observed.

![Figure 5.2. Beacon delivery ratio and UDP throughput at different distance from the RSU.](image)

The simple experiment demonstrates the potential of using a relay to improve the client access range and throughput. First, within a specific range from the RSU, stable and high single hop throughput can be achieved. Outside of this range, the throughput will be significantly reduced. However, the two hop relay solution can still achieve high and stable throughput. Therefore, finding a relay node inside this range may greatly improve the RSU access duration. Second, the abrupt throughput drop is related to the beacon delivery ratio. Thus, we can use the beacon delivery information to effectively determine the location to switch between direct access and using relay.

In the above experiment, the client laptop is placed stationary at each testing location, which is different from the real drive-thru scenario. However, it has been shown in the existing field experiments [33, 32, 31] that vehicle speed has little effect on the packet delivery rate and throughput at a given distance to the RSU. So we believe our experiment, though based on static scenario, still provides an adequate base for further proposing our scheme.


5.3 The Vehicle-to-Vehicle Relay (V2VR) Scheme

5.3.1 Assumptions

We assume that GPS is available in every vehicle to report the location. We also assume that each vehicle is equipped with two 802.11 wireless interfaces: one is pre-configured to infrastructure mode and the other is in ad hoc mode. We believe multiple wireless interfaces will be common for vehicles, since many applications require the infrastructure mode and the ad hoc mode to both be active and the cost of a wireless interface is low. For instance, when a roadside RSU disseminates road hazard emergency information, vehicles rely on the infrastructure mode to receive the data. At the same time, ad hoc mode may be used to propagate emergency warning to drivers behind a vehicle (or incident) to avoid multi-car collisions.

5.3.2 Scheme Overview

Figure 5.3 depicts the basic idea of our V2VR scheme. V2VR allows vehicles to establish the relay connection before entering the RSU coverage area. When a vehicle wants to extend its RSU access time on the road, it tries to find two vehicles as proxies; one in front of itself (forward proxy) and the other behind itself (backward proxy). The vehicle can establish a connection with a proxy through the wireless interface in ad hoc mode. After a link to the proxy is established, the vehicle may use the forward proxy to relay its traffic to a RSU before entering the RSU coverage or at the fringe of the RSU coverage area. Similarly, when a vehicle is leaving the RSU coverage area and its connection with the RSU becomes poor, it uses the backward proxy to relay traffic. To avoid the overhead of frequently changing proxies, the vehicle attempts to find vehicles with similar mobility to keep the relay connection for an extended period of time.

Our proposed V2VR scheme has three steps. In the first step, a client vehicle\(^1\) finds a group of proxy vehicles with similar mobility as itself, and registers with

\(^1\)A vehicle attempting to upload or download data through a RSU is called a client vehicle, or a client; a vehicle that is willing to relay traffic for other client vehicle is called a proxy vehicle, or a proxy.
them. The client vehicle will only ask vehicles in this group to act as relays because its connection with the selected group is more reliable than other vehicles. The second step occurs when a proxy vehicle enters a RSU’s coverage area; the proxy will select one client registered with itself to relay its traffic. This is called the forward relay scheme. Forward relay enables a client vehicle to access the RSU earlier and improves the client’s data throughput. It also allows a client to seamlessly switch to the infrastructure mode once a reliable connection to a RSU is established. The third step occurs when a client vehicle is leaving the RSU’s coverage area and the connection becomes poor. The client finds a proxy behind itself to relay traffic backward to the RSU, which is called the backward relay scheme.

A vehicle can determine when it has a reliable connection to the RSU based on the beacon message delivery ratio. In 802.11, the beacon rate is specified in the header of the beacon packet. Thus, a vehicle can simply count the number of beacon packet it hears during a time frame, and compute the delivery ratio. The area with high beacon delivery ratio implies a vehicle can reliably communicate with the RSU, e.g. within 60 meters in our experiment (see Figure 5.2(b)). The vehicle can also find the location where the delivery ratio has an abrupt drop, which is called the RSU optimal access range (AOR).

It is well recognized in traffic flow theory that vehicles in the traffic are generally grouped in a “platoon pattern”, with a surge of vehicles, followed by a gap in traffic [80]. Vehicles in a group generally move with similar velocities which are determined by the leading vehicle. In a short distance (e.g. 400-600 meters where a RSU is accessible), few vehicles in a group would dramatically change their velocities, leaving the platoon group in a consistent state. In the V2VR scheme,
we propose a stochastic model to explore the above properties so that a vehicle can find a proxy in its platoon group with good probability, and the connection can remain stable for a short period of time.

5.3.3 Step 1: A Stochastic Model for Finding A Reliable Proxy Group

A client attempts to find proxies to relay its traffic. It is not practical for a client to rely on one proxy for an extended period of time due to mobility. However, frequently searching for new proxies should also be avoided because the increased overhead can offset the benefit of relay. Thus, in V2VR, a client attempts to establish reliable one-hop links with selected proxies and ensure the links do not break within a reasonable period of time, e.g. several minutes. We define this time period as **lease time** and the selected proxies as **reliable proxies**. Clients search for new, reliable proxies only after lease time expires. A reliable proxy is expected to have similar mobility as the client vehicle; vehicles may move at different speeds and in different lanes. Mixed with acceleration and deceleration, it may be difficult for the client to find a proxy with matching mobility, or to even measure the mobility similarities. Fortunately, vehicle movement on the roadway is not completely random and we are particularly interested in exploring the movement similarities for a group of vehicles in a “platoon pattern”. The movement of a vehicle in a platoon group is restricted by the vehicle ahead and behind itself, as well as the roadway speed limit. Thus, it has some well defined movement characteristics, which can help to predict the relative location of two vehicles in a platoon group after a period of time.

In this section, we propose a stochastic model to predict the relative distance of two vehicles in a platoon group based on the vehicle’s past movement profile. The result can help the client find a group of proxies with similar mobility, so that the links between them are more likely to be stable and last longer.

For vehicles moving in a platoon, each vehicle is followed by another in a lane. A roadway may have multiple lanes, and we assume that all the vehicles in the same lane travel at an identical **lane speed**, and different lanes have different lane speeds. In order to pass other vehicles, a vehicle must change to a faster lane, and
Table 5.1. Notations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>Number of lanes.</td>
</tr>
<tr>
<td>$v_i$</td>
<td>Speed of vehicles in Lane $i$.</td>
</tr>
<tr>
<td>$(i, j)$</td>
<td>Paring state. It represents that the client and proxy vehicle are in Lane $i$ and Lane $j$ respectively.</td>
</tr>
<tr>
<td>$a_{i,j}$</td>
<td>Rate at which the client vehicle change from Lane $i$ to Lane $j$.</td>
</tr>
<tr>
<td>$b_{i,j}$</td>
<td>Rate at which the proxy vehicle change from Lane $i$ to Lane $j$.</td>
</tr>
<tr>
<td>$t$</td>
<td>Lease time. Time period when a reliable proxy group is valid.</td>
</tr>
</tbody>
</table>

The speed is determined by the lane speed.

The mobility profile of a vehicle is defined by the lane changing rates of the vehicle. Let $a_{i,j}$ denote the lane changing rate of vehicle $a$ from Lane $i$ to Lane $j$. On a $n$-lane roadway, the mobility profile of the vehicle $a$ can be formally defined by Formula 5.1.

$$M = \{a_{i,j} | 1 \leq i, j \leq n\}$$  \hspace{1cm} (5.1)

The lane changing rate over a given period of time can be easily computed by the vehicle itself. A vehicle, say vehicle $a$, only needs to record the time it spent in the previous lane, every time it moves to another lane. Then it can easily compute its average time spent in Lane $i$ before moving to $j$. The reciprocal of the average time yields the value of $a_{i,j}$. If we assume the time of a vehicle staying in a lane is exponential, the process of vehicle $a$ moving from Lane $i$ to Lane $j$ can be modeled as a Poisson process with mean of $a_{i,j}$. Since vehicles can only move to the adjacent lanes, $a_{i,j}$ satisfies $a_{i,j} = 0, \forall 0 \leq i, j \leq n, |i - j| > 1$.

To study the relative movement of a client and a proxy vehicle, we propose a Markov Chain Model. The state of the Markov Chain is defined as $(i, j)$, where $i$ and $j$ represent the lane number of the client vehicle and the proxy respectively. The amount of time the process spends in state $(i, j)$ before making a transition is exponential with a mean $1/u_{(i,j)}$. $u_{(i,j)}$ is related with the rates of the client and proxy moving to their adjacent lanes. Let $a_{i,j}$ and $b_{i,j}$ represent the lane changing rate of the client and proxy respectively, $u_{(i,j)}$ can be computed by Formula 5.2.
\[ u_{(i,j)} = a_{i,i-1} + a_{i,i-1} + b_{j,j-1} + b_{j,j+1} \]  
(5.2)

When the process leaves state \((i, j)\), the probability it enters the next state \((p, q)\) is computed by Formula 5.3

\[
P_{(i,j) \rightarrow (p,q)} = \begin{cases}  
\frac{a_{ip}}{u_{(i,j)}}, & |i - p| = 1, j = q \\
\frac{b_{jq}}{u_{(i,j)}}, & |j - q| = 1, i = p \\
0, & \text{else} 
\end{cases}
\]  
(5.3)

Given this Markov Chain model, we can compute the proportion of time spent in any state \((i, j)\), denoted as \(P_{(i,j)}\). When \(i = j\), the client and proxy are moving at the same lane, so they don’t change their distances over time. However, when \(i \neq j\), the client and the proxy are moving at different lanes. Larger \(P_{(i,j)}\) implies that the two vehicles would develop more relative displacement over time. On the other hand, the difference between \(v_i\) and \(v_j\) also affects how much relative displacement it will generate after a period of time. Therefore, we use Formula 5.4 to estimate the relative displacement between the client and the proxy after a period of time \(t\).

\[
\Delta d = \sum_{1 \leq i, j \leq n} P_{(i,j)} \cdot (v_j - v_i) \cdot t
\]  
(5.4)

In order to better present the Markov model, without loss of generality, we show an example of the model on a three-lane roadway. Figure 5.4 shows the Markov chain for the lane changing states of the two vehicle, and Figure 5.5 shows the transition matrix of the Markov chain.

To obtain the proportion of time in each state \(P_{(i,j)}\), we compute the equivalent limiting probability, so we can derive the following equation system.

State (1, 1): \(P_{(1,1)}(a_{12} + b_{12}) = P_{(1,2)}b_{21} + P_{(2,1)}a_{21}\)
State (1, 2): \(P_{(1,2)}(a_{12} + b_{21} + b_{23}) = P_{(1,1)}b_{12} + P_{(1,3)}b_{32} + P_{(2,2)}a_{21}\)
State (1, 3): \(P_{(1,3)}(a_{12} + b_{32}) = P_{(1,2)}b_{23} + P_{(2,3)}a_{21}\)

\vdots
Figure 5.4. A Markov model for three-lane roadway.

\[
\begin{pmatrix}
(1,1) & a_{12} & b_{12} & 0 & 0 & \cdots & 0 \\
(1,2) & b_{21} & a_{12} & b_{23} & b_{32} & 0 & \cdots & 0 \\
(1,3) & 0 & b_{32} & a_{12} & b_{32} & a_{23} & \cdots & 0 \\
(2,1) & a_{32} & 0 & 0 & b_{32} & 0 & \cdots & 0 \\
(2,2) & 0 & a_{32} & 0 & 0 & 0 & \cdots & 0 \\
(2,3) & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
(3,1) & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
(3,2) & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
(3,3) & 0 & 0 & 0 & 0 & 0 & \cdots & 0
\end{pmatrix}
\]

Figure 5.5. Transition matrix for three-lane Markov model.

State (3, 3): 
\[P_{(3,3)}(a_{32} + b_{32}) = P_{(2,3)}a_{(23)} + P_{(3,2)}b_{32}\]

\[
\sum_{1 \leq i,j \leq 3} P(i, j) = 1
\]

Solving the above equation system yields the proportion of time in each state 
\((i, j)\), so the client can compute the displacement between the proxy and itself 
during the lease time. In the Markov chain model, a \(n\)-lane roadway generates 
\(n \times n\) states. In the real world, since \(n\) cannot be very large, the computational 
overhead of this model is very small.

To determine whether the proxy is a reliable proxy, the client needs to consider
the relative mobility, the distance to the proxy and the signal quality. Thus, a client only considers a vehicle as a reliable proxy when Formula 5.5 is satisfied.

\[
d + \sum_{1 \leq i,j \leq n} P(i, j) \cdot (v_j - v_i) \cdot t < R \tag{5.5}
\]

In Formula 5.5, \(d\) represents the current distance between the client and the proxy, \(t\) represent the lease time, and \(R\) represents the effective range of the wireless link in ad hoc mode.

With the criteria to determine a reliable proxy, we present the protocol for finding a reliable proxy group. Figure 5.6 shows the protocol of selecting a reliable proxy. The proxy selection algorithm requires client vehicles to request the mobility profile and location of the potential proxies. After a client vehicle broadcasts a *Proxy Discover* message, every proxy vehicle will reply with a *Proxy Advertise* message, which is a 3-tuple (*location, mobility profile, transmission power*). The client uses the transmission power from the proxy to compute the SNR by comparing with the receiving power of the message. If the SNR is below a pre-specified value, the proxy is not considered. The location and mobility profile from the proxy are used to determine whether it is reliable, using the proposed Markov chain model. All the proxies which satisfy Formula 5.5 are recorded as reliable proxies by the client, while reliable forward proxies and reliable backward proxies are differentiated based on their geographic locations relative to the client.

**Figure 5.6.** Select a relay proxy.

After the client determines the available and reliable forward and backward proxies, it sends a *Proxy Register* message to each of these proxies. The message also informs the proxies of the client’s lease time and location. The proxy then
stores an entry for the registered client and is ready to forward traffic for the client when it meets a RSU, and it sends a Proxy Accept message back to the client.

Thus, before meeting any RSUs, a client can find and register with its reliable forward proxies and backward proxies. These proxies are bound to the client and may help the client forward data traffic before the lease time expires. Since one proxy may be selected by multiple clients as their reliable proxy, a proxy also records a list of registered clients. When the proxy meets a RSU, it will select one client from the list to help forward data traffic. The binding between clients and reliable proxies are many to many instead of one to one. Thus, the first step of V2VR helps a client to determine a set of possible proxies before entering the RSU coverage, but does not force the client to connect one specific proxy. The V2VR scheme put off the selection of the actual relay proxy until the client and proxies move into a specific RSU. At that time, the real relay link between a client and a proxy can be established quickly with low overhead, and with the help of the existing client and proxy many-to-many binding. We believe this is the best way to achieve high reliability, efficiency and low overhead relay. Next, we will discuss how to establish a real relay link between a client and a specific proxy.

5.3.4 Step 2: Forward Relay

5.3.4.1 Establishing the forward relay connection

When a proxy vehicle receives the first beacon from a RSU, it attempts to connect to a client which has already registered to it. Because of mobility, a registered client may not still have a good link to the proxy or may not be able to communicate. Thus, the proxy needs to poll the registered client and see if the link is still alive and whether the link quality still meets the requirements. If not, the proxy attempts to serve the next registered client. The registered clients are first sorted as a list: initially the client that the proxy has successfully served in the last RSU coverage always comes first. The other registered clients are ordered by the client registration time, i.e., first registered client is selected first. The proxy polls the clients by the order of the sorted list. After a registered client is selected, the proxy sends a Forward Relay Available message to it and waits a short period of time (50 milliseconds in our simulation) for the client to respond. When the client receives
the message from the proxy, it first checks whether the link quality is good enough by checking the SNR with a pre-specified threshold. The client agrees to use the proxy only when it satisfies the link quality. If so, it replies with a Forward Relay Request message. Otherwise it does not send any message to the proxy. After the proxy receives the Forward Relay Request message, it replies the client with a Forward Relay Confirm message to confirm the establishment of the forward relay connection. If the client is not within the proxy communication range or the client decides not to use the proxy, the proxy will not receive any response during the time interval. Then the proxy will put the selected client to the tail of the sorted list and check the next client in the sorted list.

All proxies which have not connected to any clients turn their ad hoc mode interfaces into promiscuous mode. When they overhear the Forward Relay Confirm message, they know a client-proxy connection has already been established, so if they have the same client in their own registered client lists, they will mark this client and do not poll this client when they search their relay clients. This is very effective in reducing the redundant client polling, because the proxies behind the connected proxy may otherwise check the same client again.

Note that there is enough time for the proxy to poll multiple clients because there is delay between the proxy receives the first beacon (where it is usually at the fringe) and it gets good link quality to the RSU. The proxy has enough time to poll several clients before its link with the RSU becomes stable.

5.3.4.2 Connecting to a RSU

When a forward proxy determines it has a good channel quality with a RSU, it sends a Forward Relay Start message to its connected client. Upon receiving the message, the client can start to transfer data with the RSU through the proxy relay. While using the ad hoc interface to transmitting data, the client also keeps its infrastructure interface active, listening for beacon messages from the RSU and checking the SNR. When its infrastructure interface gets good link quality, it switches to use the infrastructure interface to directly connect to the RSU.
5.3.5 Step 3: Backward Relay

The backward relay is proposed to connect a client back to the RSU through a proxy when the client is moving away from a RSU and the direct link becomes poor. Different from the forward relay scheme, it is the client who selects the relay proxy from its reliable backward relay group in the backward relay scheme. The selection procedure is similar though. The client needs to poll its reliable backward proxies one by one until it finds a proxy with good enough link quality. The client also sorts its reliable backward proxies and stores them in a sorted list; initially the backward proxy that successfully served itself in the last RSU coverage always comes first. The other reliable backward proxies are ordered by their distances to the client at the time of client relay registration in a decreasing order, i.e., the farthest proxy is polled first. The proxy poll the proxies by the order of the sorted list. After the client selects a backward proxy, it sends a Backward Relay Request message to the proxy and waits a short period of time (0.1 seconds in our simulation) for the response. If the proxy has good link quality to both the RSU and the client (by measure SNR) and is ready for backward relay when receiving Backward Relay Request message, it replies the client with a Back Relay Confirm message to confirm the establishment of the backward relay connection. The proxy may not be within the client communication range or it is already busy forwarding traffic, e.g. it is a forward proxy for another client. Thus the proxy will not reply to the client. Without receiving the response from the proxy within the specified time interval, the client will put the selected proxy to the tail of the sorted list and check the availability of the next proxy in the sorted list. After successfully finding a backward proxy, the client will switch from the direct RSU link to the two-hop relay link to access the RSU.

Figure 5.7 compares the RSU access process under the traditional 802.11 standard and the proposed V2VR scheme.
5.4 Implementation

5.4.1 Background

To test the feasibility of our ideas, we implemented a simple relay scenario. The implementation was focused on the study of the operations and status transitions which are required to setup the relay path on the fly. The implementation was also designed to test how quickly a proxy could enable packet forwarding and inform a client of its connection to a RSU. We also wanted to determine how quickly a client could begin using the forward proxy once informed of the proxy’s connection with the RSU.

Our implementation was performed in a static environment. We used two laptops running Linux, kernel v2.4.5, with two NICs in each laptop. Our RSU was a Linksys WRT54GL running DD-WRTv23 as the operating system, with a third Linux laptop to act as the server connected directly to the RSU via a high-speed cable. Our packet capture was performed on an Apple Powerbook using Ethereal.

In order to relay packets, the ability to manipulate the relevant fields in packet headers when we forwarding packets from the VANET interface to the infrastructure interface was required. IP tables provide the capability to manipulate packet headers, known as masquerading. It also provides the rules for forwarding packets and can be enabled or disabled on the fly. To listen for our messages, we used the PCAP library to capture packets. All of our system information was retrieved and configured with the ioctl system call. We use MAC broadcasts as our message format to minimize overhead and vehicles do not necessarily know the address of
their neighbors.

To format our hello packet, which covers the roles of the proxy request and selection messages as well as the forward relay available message, we had to determine the information required for a vehicle to use another as a proxy. The minimum information is the potential proxy’s IP address, so the trail vehicle knows where to send its packets. To reduce the network overhead, we included the potential proxy’s MAC address to eliminate the need for ARP requests. Unless the potential proxy also decides to act as a DNS server, the DNS Server IP address(es) provided by the RSU are also required for accessing the Internet from the trail vehicle. If the potential proxy is associated with the RSU, it stores the DNS server information in the file /etc/resolv.conf. The same file is manipulated by the client after it receives a hello packet containing DNS information. The resulting packet format is seen in Figure 5.8.

5.4.2 Information Flow

In order to exchange information, the two vehicles need to communicate with one another. One of our assumptions is that each vehicle is equipped with two interfaces among which one is used strictly for vehicle to vehicle communication. Over this interface we are able to establish forwarding relationships.

Since a vehicle can act as either a proxy or a client at any time, it must determine which role it needs to play. Figure 5.9 is a visual representation of the entire decision making process for a vehicle. Given that the vehicles can communicate, the vehicle will also have selected its proxy as discussed in Section 5.3. The first step in the process is to determine whether or not the client needs to use the proxy,
Figure 5.9. The decision making process and information flow of the implementation. We define being associated as the traditional definition, plus having a valid IP address and an acceptable SNR. If the vehicle can verify that it is associated, it will use the RSU. Otherwise, it will use the vehicle that it chose to act as its forwarder for passing traffic to the RSU.

which requires determining whether or not the vehicle is associated with a RSU. We extend the notion of association to include having a valid IP address and a SNR above our reliability threshold. If any of these three requirements are not met, the vehicle considers itself not associated.

If the vehicle is associated, it must determine whether or not it wants to act as a proxy for other vehicles. If forwarding is not enabled and the node is willing to serve as a proxy, it enables masquerading and packet forwarding. The vehicle also removes the old default route, the one pointing at its proxy. If it does not choose to act as a relay, it simply sends the hello packet without a status change. If it has already enabled forwarding previously, that means it has already been through the decision making process and there are no changes required. The final step, which only applies when a vehicle is associated and is willing to act as a forwarder, is
Figure 5.10. The sequence of events for a client to use a proxy. The proxy broadcasts its hello packets and goes through a typical association process with a RSU (1 – 15). Once association is complete, the proxy’s next broadcast includes DNS information (16). After receiving status change and DNS address, the client uses the proxy to relay traffic (17 – 22).

to retrieve the number of DNS servers the RSU provided for inclusion in the hello packet.

If the vehicle fails any of the association checks, it must use the information provided by its selected proxy to configure a default route to forward traffic through the proxy. If the client has already configured the default route, nothing is changed and the node only prepares its hello packet for transmission. However, if the node has not configured itself to leverage the proxy, it must configure a default route and install an entry in its ARP cache.

The node also checks to see whether or not the proxy is able to forward traffic at this time. If so, then the node will use the DNS information contained in the hello packet to configure its own DNS entries. If the proxy is not associated or DNS is already configured, then nothing is done except packet transmission.

In preparing the packet, a node includes its own status of being an eligible proxy, its own IP, MAC, and if associated, DNS server information.

To perform these checks and transmit hello packets, we used a 100 ms interval. The interval is equivalent to the default beacon interval for our RSU, which allows the node time to attempt to associate, or go through the association process, and to update neighbors in a timely fashion. In our full scheme, we would not perform all of these checks at a specific interval, except the association check. This would
cut down on transmission overhead by reducing unchanged updates.

5.4.3 Experimental Results

The two laptops were configured to communicate with one another prior to either associating with the RSU. The lead vehicle sends hello packets to the trail vehicle, who selects its forwarder based on the fact that it only hears hello packets from one neighbor. The hello packets are packets 1, 15, and 16 in Figure 5.10. The packets were sent at 100 millisecond intervals, but were removed to save space. To simulate movement, the potential proxy associates with the RSU and requests an IP address, as seen in packets 2 – 14 in Figure 5.10. Once the lead vehicle fully associates with the RSU, it must note its status change, include the DNS server information in its hello packet, and send the information to the trail vehicle. The inclusion of a single 4-byte DNS server can be seen in the size increase from packet 15 to 16. It also shows that the lead vehicle spends less than 100 milliseconds (between two hello packets 15 and 16) to configure itself and get ready to forward traffic. Upon receiving the change in status information, the trail vehicle is able to install the DNS information and can begin accessing the Internet via the lead vehicle in under four-tenths of a second; the trail vehicle must write to its /etc/resolv.conf file, which accounts for the delay in sending its first ICMP request. The success of the forwarding can be seen in the ping sequence of packets 17 – 22. The trail vehicle sends a ping request to its gateway, which acts as the intermediary for the request and response.

The next step is to associate the trail vehicle with the RSU and for it to halt sending traffic via the lead vehicle. The trail vehicle must go through the same association process. Once the trail vehicle is fully associated with the RSU, it removes the old default route that pointed at the lead vehicle and is able to immediately begin forwarding traffic directly through the RSU.
5.5 Performance Evaluations

5.5.1 Simulation Setup

To evaluate the performance of the proposed scheme, we implemented it in ns-2 (version 2.30) to compare it with the generic 802.11-based RSU access scheme (No Relay). We simulate a simple scenario where 150 vehicles pass a RSU on an one-way road, traveling at speed of 45 miles/hour (20 m/s). The vehicles move into the road randomly, following an exponential distribution with a mean of 0.1 to 2 vehicles/second; equivalent to a mean inter-vehicle space of 10 to 200 meters. Among these 150 vehicles, 10%-50% vehicles are randomly selected to generate network traffic. Each selected vehicle initiates an FTP session and sends data to the RSU via TCP (Reno) immediately after associated with the RSU. The vehicle continues sending data until it loses connection with the RSU.

Table 5.2. Simulation Setup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation area</td>
<td>$2000m \times 500m$</td>
</tr>
<tr>
<td>Number of vehicles</td>
<td>150</td>
</tr>
<tr>
<td>Vehicle coming rate</td>
<td>0.1 - 2 vehicles/second</td>
</tr>
<tr>
<td>Vehicle velocity</td>
<td>45 miles/hour</td>
</tr>
<tr>
<td># of TCP senders</td>
<td>15 - 75</td>
</tr>
<tr>
<td>Multi-NIC setting</td>
<td>2 radios/5 channels</td>
</tr>
<tr>
<td>Transport layer module</td>
<td>TCP Reno</td>
</tr>
<tr>
<td>MAC layer module</td>
<td>802.11b</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>2 Mbps</td>
</tr>
<tr>
<td>Data packet size</td>
<td>1440 byte</td>
</tr>
<tr>
<td>RSU beacon interval</td>
<td>100 ms</td>
</tr>
<tr>
<td>RSU optimal range (AOR)</td>
<td>60 meters</td>
</tr>
</tbody>
</table>

To emulate the link quality of a real environment, the simulation parameters are setup based on our experimental results. First, we tune the transmission power level in the simulator to provide 110 meter transmission range, matching our experimental results where no beacon packet is received farther than 110 meters. Second, the packet loss ratio at different distance from the RSU is derived from our experimental results as shown in Figure 5.2, and we set the RSU access range (AOR) to 60 meters. In our scheme, each vehicle has two NICs operating at
different channels. Since the standard ns-2 does not support multi-channel, we add the two-NIC with multi-channel functionality based on the techniques provided in [81]. All the simulation parameters are shown in Table 5.2.

5.5.2 Simulation Results

5.5.2.1 RSU Connection Duration

![Graph](image)

**Figure 5.11.** Complementary CDF of the connection duration. The mean duration for the no-relay scheme and the relay scheme is 6.06 and 9.68 respectively.

Figure 5.11 shows the complementary CDF of the connection duration between the RSU and drive-thru client vehicles. The connection time is taken as the whole period during which a vehicle’s TCP packet can be successfully delivered to the RSU. In 802.11, the connection starts when a vehicle completes its association with the RSU. When the vehicle receives the last TCP ACK message from the RSU, the connection ends. In V2VR, if a proxy is found, the connection for the client starts when it successfully establishes a relay path to the RSU, i.e., receiving the Forward Relay Confirm message from the proxy; otherwise, it waits until association finishes. As shown in the figure, V2VR significantly extends the connection time for moving vehicles. The average connection duration of our relay scheme is 9.68 second, which is 60% longer than that of the no-relay scheme (6.06 s). The extended connection time is obtained by both forward and backward relay. From Figure 5.11, we can see that the connection duration in the no-relay scheme is
clustered around 6-7 seconds. During this time, the vehicle should be able to move 120-140 meters when traveling at 20 m/s. This implies that even if the vehicles can receive the RSU beacon outside the AOR, they have little time to make use of the direct link with the RSU. The connection durations of our relay scheme are more evenly distributed, since they rely on the opportunistic connection with the proxy vehicles.

### 5.5.2.2 Amount of Data Delivered

Figure 5.12 shows the average amount of data delivered by each vehicle as the number of vehicles with TCP traffic increases. When the data traffic going through the RSU is light, our relay scheme can deliver much more data than the non-relay scheme, e.g. 90% more data when 10% of the vehicles are sending TCP traffic. This is due to the extended connection time and the improved link quality.

![Graph showing the amount of data transmitted per vehicle](Figure 5.12. Average TCP traffic transmitted per vehicle)

When more vehicles are sending data, less amount of data can be delivered by each vehicle in both schemes because all vehicles share the bandwidth. As the number of TCP senders increase, the improvement on the amount of data delivered between the two schemes becomes less pronounced, and will eventually converge. These results are not surprising when considering the fact that when the RSU is saturated it cannot support more traffic even if vehicles are connected to the RSU with our relay scheme.
5.5.2.3 Data Transmission Throughput

Figure 5.13 shows the TCP throughput that a client vehicle can obtain at different distances from the RSU. In Figure 5.13, the location of the RSU is taken as the reference point 0. The negative location value represents the area where the client moves towards the RSU but has not reached it yet, while the positive location represents where the client has passed the RSU. The figure shows the results when 10%, 25% and 50% of the vehicles are uploading/downloading data in (a), (b) and (c) respectively.

The results show V2VR can extend the stable throughput connection to the RSU regardless of the client density, but it is more effective when the density is low. When a high percentage of the vehicles are simultaneously transferring data, the clients accessing the RSU through relays compete for use of the channel with the one hop clients. Each client obtains lower throughput, which may offset the
benefit of the extended connection time. When the client percentage is low, V2VR allows the client to obtain stable throughput much earlier.

We also notice that the stable connection area of 802.11 is skewed in the direction of vehicles leaving the RSUs range. The reason is that a client starts to send TCP data packets immediately after it associates with the RSU, but its link quality is often still poor. We see frequent TCP timeouts and the TCP slow start is launched. When a client achieves good link quality, it is unable to achieve a high throughput due to TCP slow start. Since the link quality is only good in a limited time duration, 802.11 is not efficient in making use of the bandwidth.

5.6 Conclusions

In this chapter we proposed a relay-based solution (called V2VR) to extend the service range of roadside RSUs. When the link quality between a drive-thru vehicle and the RSU is poor, a relay with good link quality to the vehicle and the RSU is chosen to improve the performance. We designed techniques to select forward and backward proxies based on the mobility pattern of the vehicle. We also developed a viable prototype to address the implementation issues. Experimental results and simulation results show that a significant number of vehicles can transmit much more data with our relay scheme than without relay.
Chapter 6

Cooperative Caching in Vehicular Infrastructure Mesh Networks

6.1 Introduction

Many large scale metro mesh network deployments are planned and underway in cities across the world. The generic form of metro mesh network is two-tier: mesh nodes forward data to and from wired gateway nodes along a wireless backhaul tier and provide access to associated users via a wireless access tier.

In VANETs, a single roadside unit can only provide service in a very limited geographic area. With recent advances in wireless mesh networking, multiple roadside units can be wirelessly connected and provide broader coverage. This model is called vehicular infrastructure mesh network (or mesh network for short). In such a mesh network, the participating roadside units act as mesh base stations (or mesh routers, mesh nodes), which perform routing and configuration functionalities among themselves. A vehicle can access the data by connecting to any mesh node and those mesh nodes provide backbone for disseminating data to vehicles. A mesh node does not necessarily have to store the data locally to serve vehicles. When a vehicle requests a data from one mesh node, the data can be routed from the source to the requested mesh node over the mesh backbone, and then deliver to the vehicle. Typically, two types of radios are used in the mesh nodes: one for backbone communication among mesh nodes and the other for short commu-
nication between vehicles and mesh nodes. The mesh backbone communication is usually established using long range communication techniques.

Because of the interest locality [82], many vehicles close to one roadside unit often share common interests, and request similar data. If one vehicle has accessed a data item from the roadside unit, it is quite possible that nearby vehicles access the same data some time later. It will save a large amount bandwidth and time if later accesses to the same data are served by the nearby mesh nodes who has the data instead of the faraway data source. For example, in vehicular infrastructure mesh network, if one vehicle has downloaded a movie trailer video clip from a mesh node near a theater, with high possibility, the same clip will be requested again shortly by other vehicles passing by the same mesh node.

Therefore if nodes are able to collaborate with each other, bandwidth and power can be saved, and delay can be reduced. Actually, cooperative caching [83, 84, 85, 86], which allows the sharing and coordination of cached data among multiple nodes, has been applied to improve the system performance in wireless ad hoc and mesh networks. However, these techniques [83, 84, 85, 86] are only evaluated by simulations and studied at a very high level, leaving many design and implementation issues unanswered.

There have been several implementations of wireless ad hoc routing protocols. In [87], Royer and Perkins suggested modifications to the existing kernel code to implement AODV. By extending ARP, Desilva and Das [88] presented another kernel implementation of AODV. Dynamic Source Routing (DSR) [10] has been implemented by the Monarch project in FreeBSD. This implementation was entirely in-kernel and made extensive modifications in the kernel IP stack. In [89], Barr et al. addressed issues on system-level support for ad hoc routing protocols. In [90], the authors explored several system issues regarding the design and implementation of routing protocols for ad-hoc networks. They found that the current operating system was insufficient for supporting on-demand or reactive routing protocols, and presented a generic API to augment the current routing architecture. However, none of them has looked into cooperative caching in mesh networks.

Although cooperative cache has been implemented by many researchers [91, 92], these implementations are in the Web environment, and all these implementations
are at the system level. As a result, none of them deals with the multiple hop routing problem, and can not address the on-demand nature of the mesh routing protocols. To realize the benefit of cooperative cache, intermediate nodes along the routing path need to check every passing-by packet to see if the cached data match the data request. This certainly cannot be satisfied by the existing ad hoc routing protocols.

In this chapter, we present our design and implementation of cooperative cache in vehicular infrastructure mesh networks. Through real implementations, we identify important design issues and propose an asymmetric approach to reduce the overhead of copying data between the user space and the kernel space, and hence to reduce the data processing delay.

Another major contribution of this chapter is to identify and address the effects of data pipeline and MAC layer interference on the performance of caching. Although some researchers have addressed the effects of MAC layer interference on the performance of TCP [93] and network capacity [94], this is the first work to study this problem in the context of cache management. We study the effects of different MAC layers such as 802.11 based mesh networks and multi-interface multi-channel (MIMC) based mesh networks, on the performance of caching. Our results show that the asymmetric approach outperforms the symmetric approach in traditional 802.11 based mesh networks by removing most of the processing overhead. In MIMC mesh networks, the asymmetric approach can significantly reduce the data access delay compared to the symmetric approach due to data pipelines.

The rest of the chapter is organized as follows. Section 6.2 presents our design and implementation of cooperative cache for vehicular infrastructure mesh networks. In Section 6.3, we present our prototype and experimental results. Section 6.4 extends our cooperative cache design to a large scale network and presents extensive simulation results based on various MAC layers. Section 6.5 concludes the chapter.
6.2 Design and Implementation of Cooperative Caching

In this section, we first present the basic ideas of the three cooperative caching schemes proposed in [86]: CachePath, CacheData, and HybridCache. Then, we discuss some design issues, and present our design and implementation of cooperative cache in vehicular infrastructure mesh networks.

![Figure 6.1. A wireless mesh infrastructure network.](image)

6.2.1 Cooperative Caching Schemes

Figure 6.1 illustrates the CachePath concept. Suppose node $N_1$ requests a data item $d_i$ from $N_0$. When $N_3$ forwards $d_i$ to $N_1$, $N_3$ knows that $N_1$ has a copy of the data. Later, if $N_2$ requests $d_i$, $N_3$ knows that the data source $N_0$ is three hops away whereas $N_1$ is only one hop away. Thus, $N_3$ forwards the request to $N_1$ instead of $N_4$. Many routing algorithms (such as AODV [12] and DSR [10]) provide the hop count information between the source and destination. Caching the data path for each data item reduces bandwidth and power consumption because nodes can obtain the data using fewer hops. However, mapping data items and caching nodes increase routing overhead, and the following techniques are used to improve CachePath’s performance.

In CachePath, a node need not record the path information of all passing data. Rather, it only records the data path when it is closer to the caching node than the data source. For example, when $N_0$ forwards $d_i$ to the destination node $N_1$ along the path $N_5 - N_4 - N_3$, $N_4$ and $N_5$ won’t cache $d_i$ path information because they are closer to the data source than the caching node $N_1$. In general, a node caches
the data path only when the caching node is very close. The closeness can be defined as a function of the node's distance to the data source, its distance to the caching node, route stability, and the data update rate. Intuitively, if the network is relatively stable, the data update rate is low, and its distance to the caching node is much shorter than its distance to the data source, the routing node should cache the data path.

In CacheData, the intermediate node caches the data instead of the path when it finds that the data item is frequently accessed. For example, in Figure 6.1, if both $N_6$ and $N_7$ request $d_i$ through $N_5$, $N_5$ may think that $d_i$ is popular and cache it locally. $N_5$ can then serve $N_4$’s future requests directly. Because the CacheData approach needs extra space to save the data, it should be used prudently. Suppose $N_3$ forwards several requests for $d_i$ to $N_0$. The nodes along the path $N_3$, $N_4$, and $N_5$ may want to cache $d_i$ as a frequently accessed item. However, they will waste a large amount of cache space if they all cache $d_i$. To avoid this, CacheData enforces another rule: A node does not cache the data if all requests for the data are from the same node.

In this example, all the requests $N_5$ received are from $N_4$, and those requests in turn come from $N_3$. With the new rule, $N_4$ and $N_5$ won’t cache $d_i$. If $N_3$ receives requests from different nodes, for example, $N_1$ and $N_2$, it caches the data. Certainly, if $N_5$ later receives requests for $d_i$ from $N_6$ and $N_7$, it can also cache the data.

CachePath and CacheData can significantly improve system performance. Analytical results [86] show that CachePath performs better when the cache is small or the data update rate is low, while CacheData performs better in other situations. To further improve performance, we can use HybridCache, a hybrid scheme that exploits the strengths of CacheData and CachePath while avoiding their weaknesses. Specifically, when a node forwards a data item, it caches the data or path based on several criteria discussed in [86].

### 6.2.2 Design Issues on Implementing Cooperative Cache

To realize the benefit of cooperative cache, intermediate nodes along the routing path need to check every passing-by packet to see if the cached data match the data
request. This certainly cannot be satisfied by the existing mesh routing protocols. Next, we look at two design options.

### 6.2.2.1 Integrated Design

In this option, the cooperative cache functionalities are integrated into the network layer, so that the intermediate node can check each passing by packet to see if the requested data can be served. Although this design sounds straightforward, several major drawbacks make it impossible in real implementation.

The network layer is usually implemented in kernel, and hence, the integrated design implies a kernel implementation of cooperative cache. However, it is well known that kernel implementation is difficult to customize and then it is difficult to handle different application requirements. Secondly, kernel implementation will significantly increase the memory demand due to the use of CacheData. Finally, there is no de facto routing protocol for mesh networks currently. Implementing cooperative cache at the network layer requires these cache and routing modules to be tightly coupled, and the routing module has to be modified to add caching functionalities. However, to integrate cooperative cache with different routing protocols will involve tremendous amount of work.

### 6.2.2.2 Layered Design

The above discussions suggest that a feasible design should have a dedicated cooperative cache layer resided in the user space; i.e., cooperative cache is designed as a middleware lying right below the application layer and on top of the network layer (including the transport layer).

There are two options for the layered design. One naive solution uses cross-layer information, where the application passes data request (search key) to the routing layer, which can be used to match the local cached data. However, this solution not only violates the layered design, but also adds significant complexity to the routing protocol which now needs to maintain a local cache table. Further, if an intermediate node needs to cache the data based on the cooperative cache protocol, it has to deal with fragmentation issues since some fragments of the data may not go through this node. Thus, this naive solution does not work in practice.
Another solution is to strictly follow the layered approach, where the cooperative cache layer is on top of the network layer (TCP/IP). Figure 6.2 shows the message flow (dashed line) in the layered design. In the figure $N_5$ sends a request to $N_0$. Based on the routing protocol, $N_5$ knows that the next hop is $N_4$ and sends the request to $N_4$ encapsulating the original request message. After $N_4$ receives the request, it passes the message to the cache layer which can check if the request can be served locally. This process continues until the request is served or reaches $N_0$. After $N_0$ receives the request, it forwards the data back to $N_5$ hop by hop, which is the reverse of the data request, as shown in Figure 6.2 (b). Note that the data has to go up to the cache layer in case some intermediate nodes need to cache the data.

Although this solution can solve the problems of the naive solution, it has significant overhead. For example, to avoid caching corrupted data, reliable protocols such as TCP are needed. However, this will significantly increase the overhead, since the data packets have to move to the TCP layer at each hop. Note that the data packets only need to go to the routing layer if cooperative cache is not used. Further, this solution has a very high context switching overhead. At each
intermediate node, the packets have to be copied from the kernel to the user space for cache operations, and then re-injected back to kernel to be routed to the next hop.

**The pipeline effect:** Another problem of the layered design is the lack of data pipeline. Normally, the transport layer can fragment a large data item into many small data packets, which are sent one by one to the next hop. If there are multi-hops between the sender and the receiver, these small packets can be pipelined and the end-to-end delay can be reduced.

In cooperative cache, the caching granularity is at the data item level. Although a large data item is still fragmented by the transport layer, there is no pipeline due to the layered design. This is because the cache layer is on top of the transport layer, which will reassemble the fragmented packets. Since all packets have to go up to the cache layer hop by hop, the network runs like “stop and wait” instead of “sliding window”. This will significantly increase the end-to-end delay, especially for data with large size.

### 6.2.3 The Asymmetric Cooperative Cache Approach

To address the problem of the layered design, we propose an asymmetric approach. We first give the basic idea and then present the details of the scheme.

#### 6.2.3.1 The Basic Idea

In our solution, data requests and data replies are treated differently. The request message still follows the path shown in Figure 6.2 (a); however, the reply message follows a different path. If no intermediate node needs to cache the data, $N_0$ sends the data directly to $N_5$ without going up to the cache layer. Suppose $N_3$ needs to cache the data based on the cooperative cache protocol, as shown in Figure 6.3. After $N_3$ receives the request message, it modifies the message and notifies $N_0$ that the data should be sent to $N_3$. As a result, the data are sent from $N_0$ to $N_3$ through the cache layer, and then sent to $N_5$. Note that the data will not go to the cache layer in intermediate nodes such as $N_1$, $N_2$, and $N_4$ in this example. In this way, the data only reach the routing layer for most intermediate nodes, and go up to the cache layer only when the intermediate node needs to cache the data.
Although the request message always needs to go up to the cache layer, it has a small size, and the added overhead is limited.

![Diagram](image)

**Figure 6.3.** In the asymmetric approach, the data reply only goes up to the cache layer at the intermediate nodes that need to cache the data.

If the requested data item is large, this asymmetric approach allows data pipeline between two caching nodes, and hence reduces the end-to-end delay. The cache layer processing overhead, especially data copying between kernel and user spaces, is also minimized because the data item is not delivered to the cache layer at the nodes that are unlikely to cache the data. Next, we discuss the details of our asymmetric approach.

### 6.2.3.2 The Asymmetric Approach

Our asymmetric caching approach has three phases.

**Phase 1: Forwarding the Request Message:** After a request message is generated by the application, it is passed down to the cache layer. To send the request message to the next hop, the cache layer wraps the original request message with a new destination address which is the next hop to reach the data server (real destination). Here, we assume that the cache layer can access the routing table and find out the next hop to reach the data center. This can be easily accomplished if the routing protocol is based on DSR or AODV. In this way, the packet is received and processed hop by hop by all nodes on the path from the requester to the data server.

For example, in Figure 6.2 (a), when $N_5$ requests $d_i$ from $N_0$, it adds a new header where the destination of the data request becomes $N_4$, although the real destination should be $N_0$. After $N_4$ receives and processes the packet, it changes the destination to be $N_3$, and so on, until the request packet arrives at $N_1$. 
When an intermediate node receives the request message and delivers to the cache layer, the cache manager performs two tasks: First, it checks if it has the requested data in its local cache; if not, it adds its local information to the request packet. The local information includes the access frequency of the requested data, channel used and throughput of the link where the request is received. Its node id will also be added to Path_List, which is a linked list encapsulated in the cache layer header. When the request message reaches the node who has the data, Path_List in the message will include all the intermediate nodes along the forwarding path.

**Phase 2: Determining the Caching Nodes:** When a request message reaches the data server (the real data center or the intermediate node that has cached the requested data), the cache manager decides the caching nodes on the forwarding path, which will be presented in Section 6.2.3.3. Then the ids of these caching nodes are added to a list called Cache_List, which is encapsulated in the cache layer header.

**Phase 3: Forwarding the Data Reply:** Unlike the data request, the data reply only needs to be processed by those nodes that need to cache the data. To deliver the data only to those that will cache the data, tunneling techniques [95] are used. The data reply is encapsulated by the cache manager, and tunneled only to those nodes appeared in Cache_List. As shown in Figure 6.3, suppose the intermediate node N_3 needs to cache data d_i. Then, N_3 and N_5 are the nodes to process the data at the cache layer. N_0 includes N_3 and N_5 in the cache header of the data item d_i, and first sets the destination address of d_i to be N_3. When N_3 receives any fragmented packet of d_i, the routing layer of N_3 will deliver the packet upward to transport layer and then to the cache layer. After the whole data item d_i has been received by N_3, it caches the data item, sets the next destination using the next entry in Cache_List, which is N_5, and then passes the data down to the routing layer. After N_5 receives the data, it delivers it to the application layer.

### 6.2.3.3 Determining the Caching Nodes

When a request reaches the data server, the cache manager examines the Path_List, and determines which nodes in the Path_List to cache the data. One advantage of letting the data server make the decision is because the data server can use global
information to achieve better performance.

The data server needs to measure the benefit of caching a data item on an intermediate node and use it to decide whether to cache the data. After an intermediate node ($N_i$) caches a data item, $N_i$ can reply later requests using the cached data, instead of forwarding the requests to the data server, saving the communication overhead between $N_i$ and the data center. However, caching data at $N_i$ increases the delay of returning the data to the current requester, because it adds extra processing delay at $N_i$, and the data reassembly at $N_i$ may affect possible pipelines.

Suppose the data server ($N_0$) receives a request from a client ($N_n$). The forwarding path ($N_0, N_1, \cdots, N_{n-1}, N_n$) consists of $n-1$ intermediate nodes, where $N_i$ is the $i$th node from $N_0$ on the path. To compute the benefit of caching a data item, the data server needs the following parameters.

- **Excluded data access frequency ($f_i$):** the data access request frequency for a given data item received by $N_i$, but excluding the requests forwarded from other nodes on the forwarding path. Each node can easily measure the total request frequency ($a_i$) it receives for the given data item, including the requests generated by itself, and those forwarded from its neighbors. They can attach this information to the forwarding request, and the data server can compute $f_i$ as

  \[
  f_i = \begin{cases} 
  a_i - a_{i+1}, & \text{if } 1 \leq i \leq n-1 \\
  a_n, & \text{if } i = n
  \end{cases}
  \]

- **Tunneling delay ($d_{i,j}(S)$):** the delay to forward a data item with size $S$ from the cache layer of $N_i$ to the cache layer of $N_j$, without handing the data up to the cache layer of any intermediate nodes. $d_{i,j}(S)$ is hard to measure because it is affected by many factors, such as transmission interference, channel diversities, node mobility, etc.. We assume $d_{i,j}(S)$ is known at this point to introduce our optimal placement model. We will present heuristics to calculate it later.

**Optimal Cache Placement**
We first define a new term called aggregate delay, which includes the time consumed to reply the current client request and the delay to reply future data requests coming from the same path. We can also assign different weights to two parts of the delay based on the optimization objective, e.g., give less weight to the future access delay if the current request has strict delay requirement. But for simplicity, we weight current and future access delay equally here in this dissertation. Below, we formally define the optimal cache placement problem.

**Definition:** Given a n-hop forwarding path \(N_0, N_1, \cdots, N_n\) where \(N_0\) is the data server and \(N_n\) is the client, the problem of optimal cache placement is to find a cache node set \(P = \{N_{c_1}, \cdots, N_{c_m}|0 < c_1 < c_2 < \cdots < c_m < n\}\), which has the minimum aggregate delay for a given period of time \(\Delta t\).

In this definition, \(c_i\) refers to the subscript of the node \(N_{c_i}\) on the forwarding path, which implies that node \(N_{c_i}\) is \(c_i\) hops away from the data server. For instance, if only \(N_2\) and \(N_4\) are selected as cache node in a forwarding path \(N_0, N_1, \cdots, N_{n-1}, N_n, c_1 = 2\) and \(c_2 = 4\).

Given a cache placement \(P = \{N_{c_1}, N_{c_2}, \cdots, N_{c_m}\}\), the aggregate delay is computed by considering both the latency to return the current rely \(L_c\), and the latency to reply future data requests \(L_f\). We have:

\[
L_c = \sum_{i=1}^{m} h(S_D) + d_{0,c_1}(S_D) + \sum_{i=1}^{m-1} d_{c_i,c_{i+1}}(S_D) + d_{c_m,n}(S_D) \tag{6.1}
\]

where \(S_D\) is the data size and \(h(S_D)\) is the data processing delay. Since the data are cached at \(m\) intermediate nodes, \(\sum_{i=1}^{m} h(S_D)\) considers the cache processing delay at these nodes. As the data are reassembled at \(m\) nodes, the forwarding path is cut into \(m+1\) pieces. The rest part of Eqn. 6.1 considers the total forwarding delay as the summation of the tunneling delays on each piece.

\[
L_f = \sum_{j=1}^{c_1-1} f_j(d_{0,j}(S_R) + d_{0,j}(S_D))\Delta t + \sum_{i=1}^{m-1} \sum_{j=c_i}^{c_{i+1}-1} f_j(d_{c_i,j}(S_R) + d_{c_i,j}(S_D))\Delta t + \sum_{j=c_m}^{n-1} f_j(d_{c_m,j}(S_R) + d_{c_m,j}(S_D))\Delta t \tag{6.2}
\]
where $S_R$ and $S_D$ are the size of the request message for the data request and the data itself respectively. Eqn. 6.2 computes the future delay by assuming that any future request going through the node on the current forwarding path will be replied by the first caching node met by the request. The replied data will be sent to the client without handing up to the cache layer of the intermediate node. Thus, the cache placement set $P$ is optimal if it can minimize the weighted aggregate delay $L$, which is given by

$$\minimize L = L_c + L_f$$  \hspace{1cm} (6.3)$$

Heuristics to calculate $d_{i,j}(S)$

We first calculate the throughput between two caching nodes $N_i$ and $N_j$, denoted as $T_{i,j}$. Each node passively estimates its link throughput to its one hop neighbors. $N_i$ can estimate the throughput to $N_{i+1}$ (i.e. $T_{i,i+1}$) by using the request message size divided by the link transmission delay. The multihop throughput is computed recursively by considering the node interference and channel diversity on the forwarding path. Let $l_i$ denote the channel used between $N_i$ and $N_{i+1}$, and assume that the interference range is $p$ hops. $T_{i,j+1}$ ($j > i$) is computed as:

$$T_{i,j+1} = \begin{cases} 
\min\{T_{i,j}, T_{j,j+1}\}, & \text{if } l_j \neq l_{j-1}, l_{j-2}, \ldots, l_{j-p} \\
1/(T_{i,j} + \frac{1}{T_{j,j+1}}), & \text{otherwise}
\end{cases}$$ \hspace{1cm} (6.4)$$

In Formula 6.4, if the channel used by link $l_j$ is different from the links ($l_{j-1}, \ldots, l_{j-p}$) within its interference range, link $l_j$ can transmit the packet concurrently with any of those links. So after adding link $l_j$, the end-to-end throughput can be better sustained, which is given by the link throughput of $l_j$ or the already obtained throughput, whichever is lower. Otherwise, link $l_j$ interferes with the previous added links, and reducing the throughput.

Given $T_{i,j}$ computed by Equation 6.4, and $B$ as the size of MTU, $d_{i,j}(S)$ can be computed as:

$$d_{i,j}(S) = \left\lceil \frac{S}{B} \right\rceil \cdot B \cdot \frac{1}{T_{i,j}} + \sum_{k=i}^{j-1} \frac{B}{T_{k,k+1}}$$ \hspace{1cm} (6.5)$$

Formula 6.5 considers that a data item may be fragmented to multiple packets, with the maximum packet size as $B$. In the formula, the first term of the summation approximately computes the waiting time for the last packet to be sent, which is
after all the previous packets being injected to the forwarding path. The second term estimates the delay to transmit the last packet from the data server to the client.

When a forwarding node \( N_i \) receives a request message from the link, it can attach the link throughput \( T_{i,i+1} \) and the channel used on this link \( l_i \) to the request packet header and keeps forwarding. So given a data item of size \( S \), based on the link channel and throughput information, the data server can compute \( d_{i,j}(S) \) \((0 \leq i,j \leq n)\) by using Equ. 6.4 and 6.5.

**A Greedy Cache Placement Algorithm**

To get the optimal cache placement, the data server needs to compute the aggregate delay for every possible cache placement set. Since there are \( 2^n \) \((n\) is the length of the forwarding path\) possible ways to choose cache placement set, it takes \( \Theta(2^n) \) time, which is too expensive. Therefore, we propose a greedy heuristic to efficiently compute the optimal cache placement.

Let \( P^*(k) \) be the optimal cache placement for a forwarding path when only the nearest \( k \) hops from the data server are considered as possible cache nodes. With the same condition, let \( L^*(k) \) be the aggregate delay of the optimal placement \( P^*(k) \), and \( p^*(k) \) be the hop distance of the farthest cache node from the data server in \( P^*(k) \).

When \( k = 0 \), no cache node is between the data server and the client, and then the data server \( N_0 \) transmits the data directly to the client \( N_n \) without reassembling at any intermediate node. All future requests from \( N_i \) need to get data from the data server. Therefore, \( P^*(0) = \emptyset \), \( p^*(k) = 0 \), and

\[
L^*(0) = d_{0,n}(S_D) + \sum_{i=1}^{n-1} f_i \cdot (d_{0,i}(S_D) + d_{0,i}(S_R)) \cdot \Delta t
\]

Given \( L^*(k) \), \( P^*(k) \) and \( p^*(k) \), we check whether to select the intermediate node \( N_{k+1} \) as a cache node. If \( N_{k+1} \) is selected, we have

\[
L(k + 1) = L^*(k) + h(S_D) + (d_{p^*(k),k+1}(S_D) + d_{k+1,n}(S_D)) - d_{p^*(k),n}(S_D) - \sum_{i=k+1}^{n-1} (d_{p^*(k),i}(S_D) + d_{p^*(k),i}(S_R)) - d_{k+1,i}(S_D) - d_{k+1,i}(S_R))f_i \Delta t
\]

If \( L(k + 1) \) is smaller than \( L^*(k) \), \( N_{k+1} \) is selected as the cache node since it
reduces the aggregate delay. The optimality is updated: $L^*(k+1) = L(k+1)$, $P^*(k+1) = P^*(k) \cup N_{k+1}$, and $p^*(k+1) = N_{k+1}$; otherwise, $P^*(k+1)$, $L^*(k+1)$, and $p^*(k+1)$ keep unchanged from $P^*(k)$, $L^*(k)$, and $p^*(k)$. Figure 6.4 shows the detail of the algorithm. The algorithm has complexity of $\Theta(n^2)$, which is much more efficient than the optimal algorithm.

6.2.4 System Implementation

The cooperative cache middleware sits below the application layer and above the network layer. It consists of three parts: Cooperative Cache Agent (CCA), Cooperative Cache Daemon (CCD), and Cooperative Cache Supporting Library (CCSL).

CCA is the module that maps application protocol messages to corresponding cooperative cache layer messages. CCD is the component that implements different cooperative cache mechanisms. CCSL is the core component to provide primitive operations of the cooperative cache, e.g., checking passing-by packets, recording data access history, and cache read/write/replacement primitives.

Figure 6.5 illustrates the software architecture of CCSL. As shown in the figure, the Cache table is used to record data access. It is a hash table keyed by data id. Data items are cached in the data cache. Besides these two components, a list of recently received requests is maintained to detect duplicate data requests. If the data request is not a duplicate, it will be passed to the Cooperative Cache Daemon (CCD). An interface is provided between CCSL and the routing daemon. It enables CCSL to get the routing information which is used for transmitting cooperative cache layer packets.

CCSL encapsulates the complex mechanisms of the cooperative cache to provide simple interfaces to CCD. For example, when a data request is issued, CCD constructs a data request packet and calls send_packet() to send it. send_packet() reads the destination address of this packet, consults routing daemon for the next hop address, and sends a packet containing the received data request to the next hop. Another example is cache_data(). When cache_data() is called by CCD, it checks the data cache for spare space and then saves the data item. If there is not enough space, cache replacement is used to find enough space.
Notations

- $P$: cache placement set.
- $L$: aggregate delay.
- $pos$: hop distance of the farthest cache node from the data server.
- $f[i]$: Excluded data access frequency on $N_i$.
- $d_D[i, j]$: delay of forwarding the data item from $N_i$ to $N_j$.
- $d_R[i, j]$: delay of forwarding the data request from $N_j$ to $N_i$.
- $S_D$: size of the data item.
- $S_R$: size of the data request.
- $R[i]$: link throughput between nodes $N_i$ and $N_{i+1}$.
- $T[i, j]$: link throughput between nodes $N_i$ and $N_j$.
- $l[i]$: channel used for the link between nodes $N_i$ and $N_{i+1}$.
- $h(S)$: cache processing cost for the data size of $S$.
- $\Delta t$: the expiration time of the data item.

Cache Placement Algorithm

1: input: $f[]$, $R[]$, $l[]$, $\Delta t$, $S_R$, $S_D$.
2: output: $P$.
3: 
4: for $i = 0$ to $n - 1$ do
5:     $T[i, i + 1] = R[i]$;
6: end for
7: 
8: for $i = 0$ to $n$ do
9:     compute $T[i, j]$, $d_D[i, j]$ and $d_S[i, j]$ using Equ. 6.4 and 6.5;
10: end for
11: 
12: $L \leftarrow d_D[0, n]$;
13: $P \leftarrow \emptyset$;
14: $pos \leftarrow 0$;
15: for $i = 0$ to $n - 1$ do
16:     $L \leftarrow L + f[i](d_D[0, i] + d_R[0, i])\Delta t$;
17: end for
18: for $k = 0$ to $n - 2$ do
19:     $\Delta L_c \leftarrow h(S_D) + d_D[pos, k + 1] + d_D[k + 1, n] - d_D[pos, n]$;
20:     $\Delta L_f \leftarrow 0$;
21: for $i = k + 1$ to $n - 1$ do
22:     $\Delta L_f \leftarrow \Delta L_f + (d_D[pos, i] + d_R[pos, i] - d_D[k + 1, i] - d_R[k + 1, i])f[i]\Delta t$;
23: end for
24: $L' \leftarrow L + \Delta L_c - \Delta L_f$;
25: if $L' < L$ then
26:     $L \leftarrow L'$;
27:     $P \leftarrow P \cup N_{k+1}$;
28:     $pos \leftarrow k + 1$;
29: end if
30: end for

Figure 6.4. The greedy algorithm for determine the cache placement.
6.3 The Prototype and Experimental Results

To evaluate the performance of the cooperative cache implementation, we set up an very small mesh network as shown in Figure 6.6. Five nodes are Dell Pentium laptops with 1.6 GHz CPU, and 512MB memory. Each Laptop has a Dell TrueMobile 802.11 wireless cards configured in ad hoc mode. The dashed circle around each node indicates the transmission range. Any two nodes connected with a solid line are direct neighbors. The routing protocol is based on AODV [12]. The implementation is based on Linux kernel version 2.4.21.
6.3.1 Experimental Results

In this section, we compare the performance of the symmetric and asymmetric cooperative cache approaches. In the symmetric approach, both data request and data reply go up to the cache layer at every intermediate forwarding node. As shown in Figure 6.6, \( N_1 \) is the data center which stores 100 test files of sizes: 0.9KB, 1.3KB, 1.9KB, 3.2KB and 6.4KB. \( N_4 \) and \( N_5 \) randomly choose files and initiate data requests.

<table>
<thead>
<tr>
<th></th>
<th>0.9KB</th>
<th>1.3KB</th>
<th>1.9KB</th>
<th>3.2KB</th>
<th>6.4KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimpleCache</td>
<td>28.56</td>
<td>31.19</td>
<td>42.12</td>
<td>60.64</td>
<td>118.26</td>
</tr>
<tr>
<td>Symmetric</td>
<td>24.87</td>
<td>27.13</td>
<td>36.97</td>
<td>49.30</td>
<td>102.38</td>
</tr>
<tr>
<td>Asymmetric</td>
<td>22.56</td>
<td>24.21</td>
<td>32.36</td>
<td>46.09</td>
<td>93.70</td>
</tr>
</tbody>
</table>

Data from Table 6.1 shows that the Asymmetric approach reduces the data access latency by 20-23% compared to non-cooperative cache (SimpleCache) and the symmetric approach reduces the data access latency by 12-18% compared to SimpleCache. This is because cooperative cache helps the requester to get data from nearby nodes when the data are not locally cached by the requester. For these two cooperative cache approaches, the asymmetric approach experiences on average 10% less data access delay compared to the symmetric approach. This delay reduction is achieved by reducing the number of times that the data item is passed from the network layer to the cooperative cache layer. In the symmetric approach, for any intermediate node, the received data item has to be passed to the cooperative cache layer which is in the user space. If this intermediate node does not need to cache the data, this context switch is not necessary. The asymmetric approach reduces the access delay by removing these unnecessary context switches.

The small scale prototype has several limitations which make in-depth performance evaluation hard to perform. First, due to the small scale, the distance between the source and the destination is short, and hence the advantage of cooperative caching is not as much as that shown in [86]. Secondly, \( N_4 \) and \( N_5 \) are the only two requesters in the network, and \( N_3 \) is the only node selected by the algorithm to do cooperative caching. A data item will be put into the cache of
Asymmetric
Symmetric
SimpleCache

Cache hit ratio

Remote cache hit
Local cache hit

Figure 6.7. Cache hit ratio of the three approaches at different data sizes.

$N_3$ after it is accessed by either $N_4$ or $N_5$ (let’s say $N_4$), and the cached data can only help $N_5$ once. Later both $N_4$ and $N_5$ can access this data from their local caches. All the cached data at $N_3$ can at most serve one request, thus the utilization of the cooperative cache is very low in this prototype. Figure 6.7 shows the cache hit ratio in our experiment, which confirms the above findings. Thirdly, since each node only has one wireless interface, due to interference, it is difficult to test the pipeline effect identified in Section 6.2.2.2. This also explains why the difference between symmetric and asymmetric approaches is relatively small, as the asymmetric approach only saves the processing delay at $N_3$.

Although our prototype has some limitations, it can be used to verify our simulation testbed which will be shown in the next section. Further, it verifies one important delay factor: the data processing delay. To better evaluate the system performance such as the pipeline effect, we collect real data from the prototype, and use the data to tune the simulation testbed to better simulate the real system performance.

<table>
<thead>
<tr>
<th>Packet type</th>
<th>Packet processing time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.9KB</td>
</tr>
<tr>
<td>Request</td>
<td>0.217</td>
</tr>
<tr>
<td>Reply</td>
<td>1.483</td>
</tr>
</tbody>
</table>

In order to accurately evaluate the processing delay at the cache layer, we use results from our prototype, which are shown in Table 6.2. The cache layer
processing delay of our simulator is strictly tuned to follow these values. The data processing delay is generally not considered in most network simulators, but it is a very important factor in our system design.

6.4 Performance Evaluations

To evaluate our design and implementation in a large scale network, and to evaluate how different factors affect the system performance, we perform extensive simulations. We also compare our approach with various design options.

6.4.1 Simulation Model

The simulation is based on ns-2 [70]. The implementation of the cooperative cache layer is ported from the real system implementation, but simplified to fit the simulator.

The MAC layer: we simulate several MAC and physical layer options, and compare the performance of various cache designs. Table 6.3 shows the complete list of MAC and physical layer options. The basic wireless interface follows 802.11b standard. The radio transmission range is 250m, and the interference range is 550m. We use the existing 802.11 MAC implementation included in the original ns-2 package as our single-interface single-channel MAC layer.

<table>
<thead>
<tr>
<th>Wireless Interface</th>
<th>Channel Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>single-interface single-channel</td>
<td>2 Mbps</td>
</tr>
<tr>
<td>multi-interface multi-channel</td>
<td>2 Mbps</td>
</tr>
</tbody>
</table>

Table 6.3. Wireless interface setup

For the multi-interface multi-channel MAC protocol, we assume each node is equipped with multiple 802.11b compatible interfaces. These interfaces can be tuned to multiple orthogonal channels [96, 97, 98, 99]. In this way, it is possible for a single node simultaneously sending and receiving packets using two independent radios, whereas neighboring nodes can also simultaneously transmit packets at other non-interfering channels. Since it is difficult to find a general MAC protocol for multi-interface multi-channel (MIMC) mesh network, we simulate it based on the MIMC mesh network architecture proposed by Raniwala et al. [99]. This is
achieved by modifying the existed 802.11 MAC protocol in ns-2; i.e., increasing the concurrent transmissions and carefully tuning the collision ratio to generate a virtual multi-interface multi-channel MAC layer which is comparable to the goodput achieved in [99].

**The client query model:** The client query model is similar to what has been used in previous studies [86]. Each node generates a single stream of read-only queries. The query generate time follows exponential distribution with mean value $T_{query}$. After a query is sent out, the node does not generate new query until the query is served. The access pattern is based on Zipf-like distribution, which has been frequently used [100] to model non-uniform distribution. In the Zipf-like distribution, the access probability of the $i^{th}$ ($1 \leq i \leq n$) data item is represented as follows.

$$P_{a_i} = \frac{1}{i^\theta \sum_{k=1}^{n} \frac{1}{k^\theta}}$$

where $0 \leq \theta \leq 1$. When $\theta = 1$, it follows the strict Zipf distribution. When $\theta = 0$, it follows the uniform distribution. Larger $\theta$ results in more “skewed” access distribution. We choose $\theta$ to be 0.8 according to studies on real Web traces [100].

**The server model:** Two data servers: server0 and server1 are placed at the opposite corners of the rectangle area. There are $n$ data items at the server side and each server maintains half of the data. Data items with even ids are saved at server0 and the rests are at server1. The data size has a range between $s_{min}$ and $s_{max}$. The data are updated only by the server. The servers serve the requests on FCFS (first-come-first-service) basis. Most system parameters are listed in Table 6.4.

We first verify our simulation testbed by configuring it with our five-node experimental topology. As shown in Table 6.5, the simulation results match that in the prototype experiment. Next we increase the scale of our network using parameters listed in Table 6.4 to collect more results.

### 6.4.2 Simulation Results

In this section, we compare the performance of the SimpleCache approach, the Symmetric Cooperative Cache (SCC) approach, and the Asymmetric Cooperative
Table 6.4. Simulation Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation area</td>
<td>$4500m \times 600m$</td>
</tr>
<tr>
<td>Number of nodes</td>
<td>100</td>
</tr>
<tr>
<td>Communication range</td>
<td>250m</td>
</tr>
<tr>
<td>Interference range</td>
<td>550m</td>
</tr>
<tr>
<td>Query generate interval</td>
<td>$T_{query} = 5s$</td>
</tr>
<tr>
<td>MTU size</td>
<td>500B</td>
</tr>
<tr>
<td>Client cache size</td>
<td>800KB</td>
</tr>
<tr>
<td>Database size</td>
<td>1000 items</td>
</tr>
<tr>
<td>Data item size</td>
<td>$s_{\text{min}} = 100B, s_{\text{max}} = 7KB$</td>
</tr>
</tbody>
</table>

Table 6.5. Simulated data access delay using the five-node topology

<table>
<thead>
<tr>
<th></th>
<th>0.9KB</th>
<th>1.3KB</th>
<th>1.9KB</th>
<th>3.2KB</th>
<th>6.4KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimpleCache</td>
<td>27.21</td>
<td>30.51</td>
<td>41.77</td>
<td>59.92</td>
<td>117.67</td>
</tr>
<tr>
<td>Symmetric</td>
<td>23.82</td>
<td>26.46</td>
<td>36.05</td>
<td>48.67</td>
<td>101.90</td>
</tr>
<tr>
<td>Asymmetric</td>
<td>11.78</td>
<td>24.03</td>
<td>32.01</td>
<td>45.09</td>
<td>94.24</td>
</tr>
</tbody>
</table>

Cache (ACC) approach in various network environments. SimpleCache is the traditional cache scheme that only caches the received data at the query node. We also compare these schemes to an Ideal Cooperative Cache (ICC) approach, which does not have processing delay at the cache layer. Further, upon receiving each packet, the cache manager makes a copy of the packet and buffers it, and then forwards the original one immediately. Thus, an intermediate node can immediately forward the packet without waiting until the whole data item is received, which can maximize the pipeline effect. It is easy to see that ICC sets up a performance upper bound that a cooperative cache scheme may achieve.

6.4.2.1 Comparisons in Traditional 802.11 based mesh Networks

Figure 6.8(a) shows the average data access delay of different designs in transitional 802.11 mesh networks. The benefit of cooperative caching is easy to see when the channel bandwidth is low (2Mbps) regardless of the cache design options. Cooperative cache increases the chance of getting data with less number of hops, and hence can reduce the access delay compared to the SimpleCache approach no matter how they are designed.

Figure 6.9 provides a closer view to compare the performance of these three
Figure 6.8. The data access delay comparison in 802.11 based mesh networks

Figure 6.9. A close view of query latency for different cooperative cache schemes in single-interface-single-channel 2M 802.11 based mesh networks

cooperative cache schemes (SCC, ACC, and ICC). As shown in the figure, the ACC approach is quite close to the performance of the ICC approach. The advantage of ACC and ICC over SCC is about 10%-25%. Based on the results of Section 6.3, most of this performance gain comes from the processing delay, but not from the pipeline effect. This is because the spacial channel reuse of 802.11 is extremely limited. In a 802.11 based mesh network, all the nodes use the same channel. When a node is transmitting a packet, the interference range can be over twice of its transmission range, forcing any other packet transmission within this distance to back-off. It has been found in [93, 94] that the best achievable spatial reuse in
general 802.11 based mesh networks is just 1/4 of the flow hop-length.

From Figure 6.8(a), we can also see that the average data access delay increases linearly to the data size. The data access delay of the SimpleCache approach significantly increases when the data size is larger than 3.5KB. This is due to network congestion. In the SimpleCache approach, each data request needs to travel more hops to be served compared to that in the cooperative cache schemes. As a result, each data request uses more network bandwidth, and the chance of network congestion is higher. In case of network congestion, the data access delay significantly increases.

By increasing the transmission rate to 5Mbps, as shown in Figure 6.8(b), the network capacity increases, and there is no network congestion in SimpleCache even when the data size increases to 7KB. From Figure 6.8(b), we can also see that the SCC approach does not have too much advantage over the SimpleCache approach. There are two reasons. First, as the data transmission rate increases, the processing overhead of the SCC approach becomes significant. Secondly, as the data transmission rate increases, it starts to have some pipelines, but the SCC approach does not allow any pipelines. The ACC approach does not have these disadvantages, and hence still has much better performance compared to the SimpleCache approach.

The Ideal Cooperative Cache (ICC) approach allows pipeline and has no processing overhead. Hence, it has the lowest data access delay. The delay of the ACC approach is quite close to optimal, which verifies that the asymmetric approach is quite effective on mitigating the cache layer overhead. It has almost the same delay as the ideal cooperative cache approach when the data size is not much larger than MTU, and there are normally not enough packets to fill in the “pipe” along the forwarding path. As the data size increases, the ACC approach has a little bit longer delay than the ICC approach, since the caching nodes stop the pipeline. But it is still much better than the SCC approach.

6.4.2.2 Comparisons in MIMC Mesh Networks

MIMC mesh network is designed to increase the bandwidth utilization and allow neighbor nodes communicate concurrently. As a result, it is easier for the nodes to take advantage of the pipeline effect. When a large data item is transmitted in
network, it is fragmented into small packets. These packets can be pipelined along
the forwarding path to maximize the throughput and reduce the data access delay.
As shown in Figure 6.10, due to data pipeline, the SimpleCache approach may
outperform the SCC approach. This is because, as discussed in the last section,
the SCC approach has high processing overhead and it is lack of pipeline.

In Figure 6.10(a), when the data size is larger than 6KB, the SimpleCache
approach still runs into severe congestion, due to excessive packets injected to the
network. As shown in Figure 6.10(b), the performance improvement of ACC over
ICC drops as the data size increases. This can be explained as follows. The major

Figure 6.10. Comparison of the data access delay in MIMC mesh networks

Figure 6.11. Comparison of the data traffic generated in 5M MIMC mesh networks
benefit of cooperative caching is to reduce the hop distance of data access. This will be translated into the reduction of data access delay in 802.11 based mesh network. However, this is not exactly true in high bandwidth MIMC mesh networks. In such networks, as long as the data item is large enough for a full pipeline, the hop distance becomes less important to the data access delay. Although caching in the intermediate node can reduce the hop distance for future data access, this delay reduction is less important. Further, it is at the cost of shortening the pipeline due to caching in the intermediate nodes. Even considering these constraints, the ACC approach outperforms the SimpleCache approach, and is very close to the ideal cooperative cache approach.

From Figure 6.10(b), we can see that the delay advantage of the cooperative cache approaches is not that significant in high bandwidth MIMC mesh networks. This is because the network has enough bandwidth to support all the nodes. However, as the nodes increase the query rate or access data of larger size, the delay of the SimpleCache becomes much higher. Similar results have been shown in Figure 6.10(a). Although the pipeline can reduce the delay, the SimpleCache approach still generates more traffic, which may result in a network congestion and longer delay. As shown in Figure 6.11, the cooperative cache schemes (ICC, SCC, ACC) generate 30-50% less data traffic than the SimpleCache approach because cooperative cache can reduce the number of hops to get the data.

6.5 Conclusions

In this chapter, we present our design and implementation of cooperative cache in vehicular infrastructure mesh networks. In our asymmetric approach, data request packets are transmitted to the cache layer on every node; however, the data reply packets are only transmitted to the cache layer on the intermediate nodes which need to cache the data. This solution not only reduces the overhead of copying data between the user-space and the kernel space, but also allows data pipeline to reduce the end-to-end delay. We have developed a prototype to demonstrate the advantage of the asymmetric approach. Since our prototype is at a small scale, we evaluate our design for a large scale network through simulations. Our simulation results show that the asymmetric approach outperforms the symmetric approach.
in traditional 802.11 based mesh networks by removing most of the processing overhead. In MIMC mesh networks, the asymmetric approach can significantly reduce the data access delay compared to the symmetric approach due to data pipelines.
Conclusions and Future Work

7.1 Summary

In this dissertation, we have designed a comprehensive solution to support data dissemination for vehicles at various locations in a metropolitan wide VANET. The solution provides several techniques which are optimized for vehicles according to their relative locations to the roadside units. We summarize these techniques in as follows.

In Chapter 3, we adopted the idea of carry and forward, where a moving vehicle carries the packet until a new vehicle moves into its vicinity and forwards the packet. Different from existing carry and forward solutions, we made use of the predictable vehicle mobility, which is limited by the traffic pattern and road layout. We proposed several vehicle-assisted data delivery (VADD) protocols: L-VADD, D-VADD, and H-VADD based on the techniques used for road selection at the intersection. Experimental results showed that the proposed VADD protocols outperform existing solutions in terms of packet delivery ratio, data packet delay and traffic overhead. Among the proposed VADD protocols, the H-VADD protocol has much better performance.

In Chapter 4, we proposed a data pouring and buffering scheme to address the data dissemination problem in densely populated areas in VANET. In Data Pouring (DP), data are periodically broadcasted along the optimally selected roads to reach the vehicles. In DP-IB, data poured from the data center are buffered and rebroadcasted at the intersections. Our simulation results showed that the
proposed DP-IB scheme can significantly improve the data delivery ratio and reduce the network traffic. Further, we provided analytical models to explore the dissemination capacity of the proposed schemes. The analytical models also provide guidelines on choosing the system parameters to maximize the dissemination capacity under different data delivery ratio requirements.

In Chapter 5, we focused on disseminating data from a roadside unit to drive-thru vehicles. We proposed a relay-based solution to extend the service range of roadside units. When the link quality between a drive-thru vehicle and the roadside unit is poor, a relay with good link quality to the vehicle and the roadside unit is chosen to improve the performance. We studied the car following pattern in a platoon, and designed techniques to select forward and backward proxies based on the mobility characteristics. We also developed a viable prototype to address the implementation issues. Experimental results and simulation results showed that a significant number of vehicles can transmit much more data with our relay scheme than without relay.

In Chapter 6, we proposed a vehicular infrastructure mesh network and presented our design and implementation of cooperative cache in such a mesh network. In our asymmetric approach, data request packets are transmitted to the cache layer on every node; however, the data reply packets are only transmitted to the cache layer on the intermediate nodes which need to cache the data. This solution not only reduces the overhead of copying data between the user-space and the kernel space, but also allows data pipeline to reduce the end-to-end delay. We developed a prototype to demonstrate the advantage of the asymmetric approach. Since our prototype is at a small scale, we evaluate our design for a large scale network through simulations. Our simulation results show that the asymmetric approach outperforms the symmetric approach in traditional 802.11 based mesh networks by removing most of the processing overhead. In multi-radio multi-channel mesh networks, the asymmetric approach can significantly reduce the data access delay compared to the symmetric approach due to data pipelines.
7.2 Future Directions

Our work to date has provided a cohesive solution for disseminating data in metropolitan wide VANETs. However, as of today, the deployment of urban VANETs is still in investigative phase; the field of building practical VANETs is still wide open. There are ample opportunities for further research in this area. Next I outline several interesting directions for future work that one could pursue.

1) Minimize bandwidth utilization with bounded delay: Different applications in VANET often call for different delay requirements. The network should maintain a low level of channel utilization, so as to ensure enough bandwidth for on time delivery of more time stringent data. Therefore, it is very important to reduce the bandwidth consumed by those more delay tolerant data. One way to achieve this is to leverage the knowledge of available traffic statistics to carefully alternate between wireless forwarding and “store and carry” strategies. For the data which do not require immediate delivery, vehicles are better to carry them longer before forwarding, as long as the data can be delivered within the required time frame. So the “store and carry” strategy is heavily utilized by the more delay tolerant data to reduce the use of wireless medium; while wireless forwarding strategy is more frequently used to forward time stringent data to reduce delivery delay. This approach can minimize communication overhead while adhering to delay constraints imposed by the application.

2) Real-time road traffic discovery: Many data dissemination schemes in VANET rely on the historical road traffic statistics to compute optimal delivery paths. But the historical data are not likely to be soon available for every roads. In addition, the historical data are not capable of capturing the real-time road traffic conditions in some special events, e.g. in the occurrence of road constructions or traffic redirection. So in some occasions, the knowledge of real-time vehicular traffic condition can greatly improve the forwarding path selection for data delivery. In order to obtain real-time road traffic condition, a path discovery scheme similar to DSR can be employed. The naive approach is to flood the route discover messages to obtain a path to the destination. But because the node connectivity in VANET is extremely unreliable due to fast vehicle movement, the path represented by forwarding nodes is of little use. A better approach is to represent the
forwarding path by recording the traversed intersections instead of the forwarding nodes. The technique presented in [101] can be used to mitigate the redundant transmissions in discovery flooding by forcing each node to hold the packet, wait for a short time and listen the channel for duplicated packet. Upon receiving the discovery reply, the source can leverage the real-time vehicular traffic information to create a road-based path consisting of successions of road intersections that have, with high probability, network connectivity among them. So by adding some reasonable overhead, this algorithm can deliver data without the knowledge of the historical traffic statistics. It can also efficiently handle the vehicle traffic dynamics in real-time.

3) Data dissemination based on data popularity: With the growing penetration of wireless enabled vehicles, pure peer-to-peer data sharing without infrastructure involved will be more and more popular. The challenge is that the peer-to-peer based VANET is an intermittent connected network. The network connectivity among vehicles is opportunistic; the connection duration is often short and unreliable. There is no guarantee whether a vehicle can get a specific data item from other nodes and how long it may take. However, if a vehicle requests a popular data item which is densely disseminated in the network, it may take much shorter time than requests a rare data item, because the chances of meeting one vehicle who has the popular data is much higher. In such a opportunistic DTN network, it is therefore a more reasonable and efficient approach to improve the accessibility of popular data than make all data accessible to users. So when a vehicle receives a data request from a neighboring node, it should return the most popular data item that is relevant to the request. A receiver is not only a data consumer, but also carries the data to serve more users with similar interests in the future. Therefore, when the sender decides which data to deliver, it should consider both the client’s current interest and overall demand in the network. The individual decision would ensure more popular data are more likely to be disseminated. Hence, the overall probability of getting useful data for every user is improved.
Proof of the Linear Equation System in Section 3.2.3

THEOREM 2. The linear equation system given by Equation 3.4 has a unique solution.

Proof:
In Equation 3.4, if $P - E$ is an $n \times n$ invertible matrix, $(P - E) \cdot X = -D$ has a unique solution given by $X = (P - E)^{-1} \cdot -D$. The rest of this section will prove that the matrix $P - E$ used in Section 3.2.3 is invertible.

It is important to relate the matrix $P - E$ to real road networks to further illustrate the properties of $P - E$. The matrix $E$ is simply an $n \times n$ identity matrix. The $n \times n$ matrix $P$ describes the system with $n$ directional roads. Note that one road with two opposite traffics is defined as two different directional roads in our model. Each row of $P$ represents a directional road, and each column represents a directional road. Most importantly, the number in the $i$th row and $j$th column of $P$ (called the $ij$th element and written $P_{ij}$) represents the probability of choosing road $j$ as the next road to forward a packet, given that the packet is currently on road $i$. Let $p_{ij}$ denote the $ij$th element in the matrix $P - E$, the following three properties of $P - E$ are useful in proving Theorem 2.

Property 1. Diagonal Property

$$p_{kk} = -1, \text{ for each } k = 1, \cdots, n.$$

Proof. If a packet is currently carried on road $k$, the next road to forward the packet cannot be itself. So the probability of selecting itself as the next road is 0. Therefore, in the matrix $P$, $P_{kk} = 0$, for each $k = 1, \cdots, n$. The values of the
diagonal elements in $P - E$ are

$$p_{kk} = P_{kk} - 1 = 0 - 1 = -1, \text{ for each } k = 1, \ldots, n.$$ 

\[\square\]

**Property 2.** Row Property - There exists at least one row $r$ in $P - E$, such that $p_{rk} = 0$, for each $k = 1, \ldots, n$ and $k \neq r$. Besides these rows, all the other rows $r'$ satisfy $\sum_{k=1,k \neq r}^{n} p_{r'k} = 1$.

*Proof.* Let’s first examine the matrix $P$. Since we assume the destination area is either within one intersection area, or at the middle of the road connecting two intersections, we can find at least one road which directly leads to the destination (without via any intermediate intersections). Let’s call this road $r$. When a packet is already carried on the road $r$, it will not be forwarded to any other road except the destination. Thus the probability of the packet reaching any other road from road $r$ is 0, i.e. $P_{rk} = 0$, for each $k = 1, \ldots, n$ and $k \neq r$. When the packet is on a road which does not directly lead to the destination (named $r'$), it may be forwarded to any of the roads directly connected with the current road with certain probability, and the summation of the probabilities of being forwarded to all these roads is $\sum_{k=1,k \neq r}^{n} P_{r'k} = 1$.

Apparently $P$ and $P - E$ have exactly the same elements except the diagonal elements. Therefore, the above properties are also hold for the matrix $P - E$. The Row Property of $P - E$ is proved. \[\square\]

**Property 3.** Column Property - At any column $c$ of the matrix $P - E$, the element $p_{kc}$ is either 0, or a positive value less than or equal to 1, for each $k = 1, \ldots, n$ and $k \neq c$.

*Proof.* In the matrix $P$, the value of the element $P_{kc}$ describes the probability of road $c$ to be chosen as the next road to forward the packet, when the packet is currently on road $k$. When road $c$ is not directly connected to road $k$, it is impossible for road $c$ to be the next road to forward the packet after road $k$, so $P_{kc}$ is equal to 0. Otherwise, the packet may be forwarded to road $c$ immediately after passing road $k$, and the probability is apparently a positive value less than or equal to 1.

Again, since $P$ and $P - E$ have exactly the same elements except for the diagonal elements, $p_{kc}$ is equal to $P_{kc}$, which is either 0, or a positive value less than or equal to 1, when $k \neq c$. \[\square\]

Let’s first simplify Equation System 3.4 by eliminating all the equations with the form

$$-x_i = -d_i.$$
The equation of this form corresponds to one row vector $P_i$ in $P$ with $p_{ij} = 0$ ($j = 1, \cdots, n$), which represents the road directly leading to the destination. We simply substitute all $x_i$ for $d_i$ in these equations in $P - E$, and call the simplified new $m \times m$ (certainly $m < n$) matrix as $A$. Apparently $A$ still holds the above three properties of $P - E$, because this simple transformation does not change any of the above properties. Also, since $A$ is reduced from $P - E$ only by using elementary row operations, to prove $A$ to be invertible is equivalent to proving $P - E$ to be invertible.

A sufficient condition to guarantee a matrix to be invertible is that this matrix is diagonally dominant and irreducible.

**DEFINITION 1.** A matrix $Q_{m \times m}$ is said to be diagonally dominant iff, for every row (or column), the sum of the absolute values of the off diagonal elements is never greater than the absolute value of the diagonal element, and at least there is one row $i$ in $Q$ such that:

$$|q_{ii}| > \sum_{k=1, k \neq i}^{m} |q_{ik}|$$

**DEFINITION 2.** A matrix $Q_{m \times m}$ is said to be irreducible iff, for any row index $i$ and column index $j$, there is always a nonnegative integer $s$ (which may be 0) and a sequence of integers $k_1, \cdots, k_s$ so that the product

$$q_{i,k_1} \times q_{k_1,k_2} \times \cdots \times q_{k_s,j}$$

is nonzero.

**LEMMA 1.** The matrix $A$ is a diagonally dominant matrix.

*Proof.* Since Property 1, 2 and 3 are held in $A$, all the values of the diagonal elements in $A$ are equal to $0 - 1 = -1$ (Property 1), and the sum of the absolute values of the off diagonal elements is less than or equal to 1 (Property 2). Further, the transformation from the matrix $P - E$ to the matrix $A$ eliminates some columns; and the eliminated columns represent the roads which directly lead to the destination. For simplicity, suppose only one column $j$ is eliminated in $P - E$, thus road $j$ is the only road directly leading to the destination. Since there must exist at least one other road $i$ (assume $i < j$, without loss of generality), which does not directly lead to the destination, but chooses road $j$ with certain probability $P_{ij}$ ($P_{ij} \neq 0$) as the next road to forward the packet (otherwise the packet cannot reach the destination when it is on road $i$). Since $P_{ij}$ is equal to the element $p_{ij}$ in the matrix $P - E$, when column $j$ in $P - E$ is eliminated, the sum of the absolute
values of the off diagonal elements in row \( i \) is reduced, and becomes less than 1. So we find one row \( i \) in the new \( (n-1) \times (n-1) \) matrix \( A \) satisfying
\[
|a_{ii}| = 1 > \sum_{k=1 \atop k \neq i}^{n-1} |a_{ik}|.
\]

When more than one columns are eliminated, this property can be proved similarly. Therefore, the matrix \( A \) is diagonally dominant.

**Lemma 2.** The matrix \( A \) is an irreducible matrix.

**Proof.** Since \( P - E \) is generated based on the real roads in a given non-partitioned area, all the roads are reachable from one to another. Thus for any two road \( i \) and \( j \), a packet can always be routed from \( i \) to \( j \) with certain probability. The only exception occurs when the packet is already on the road directly leading to the destination, and it is impossible to reach any other road. However, after we eliminate these roads in \( P - E \), and transform the matrix to \( A \), this exception does not exist in \( A \), because all the roads directly leading to the destination are eliminated. Therefore the probability of the packet routed between any pair of roads \( i \) and \( j \) is not zero. Suppose the packet is routed via the road sequence \( i, r_{k_1}, r_{k_2}, \ldots, r_{k_s}, j \). The probability of following this sequence is
\[
a_{i,k_1} \times a_{k_1,k_2} \times \cdots \times a_{k_s,j}
\]
which is not zero. Thus the matrix \( A \) is irreducible.

Since the matrix \( A \) is both diagonally dominant and irreducible, it is invertible. We conclude that the matrix \( P - E \) is also invertible, and the linear equation system shown in Equation 3.4 has a unique solution.
Bibliography


[57] “TIGER, TIGER/Line and TIGER-Related Products,” U.S. Census Bureau. URL http://www.census.gov/geo/www/tiger/


Vita
Jing Zhao

Jing Zhao was born in Kunming, China. He received the B.S. degree in Computer Science from Peking University, Beijing, China, in July 2002. He enrolled in the Ph.D. program in Computer Science and Engineering at The Pennsylvania State University in August 2003. He is a student member of IEEE.