OPPORTUNISTIC AND COOPERATIVE FORWARDING IN MOBILE AD-HOC NETWORKS WITH LIGHT-WEIGHT PROACTIVE SOURCE ROUTING

ZEHUA WANG

Opportunistic and Cooperative Forwarding in Mobile Ad-hoc Networks with Light-Weight Proactive Source Routing

by

©Zehua Wang

A thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of

Master of Engineering

Faculty of Engineering & Applied Science

Memorial University of Newfoundland

September 2011

St. John's Newfoundland

Abstract

The multi-hop wireless network has drawn a great deal of attention in the research community. Within the long period after it was proposed, the routing and forwarding operations in the multi-hop wireless network remain to be quite similar to those in the multi-hop wired network or the Internet. However, all the data transmission over the wireless medium in the wireless networks is by broadcasting in nature, which is different from the Internet. Because of the broadcast nature, many opportunities based on overhearing can be used to enhance the data transmission ability in wireless network. ExOR is the first practical data forwarding scheme which tries to promote the data transmission ability by utilizing the broadcast nature in wireless mesh networks, and the *opportunistic data forwarding* becomes a well-known term given by ExOR to name this kind of new data forwarding scheme. The basic idea in ExOR has triggered a great deal of derivations. However, almost all these derivations are proposed for wireless mesh networks or require the positioning service to support opportunistic data forwarding in the Mobile Ad-hoc Networks (MANETs). In this thesis, we propose a series of solutions to implement opportunistic data forwarding in more general MANETs, which is called Cooperative Opportunistic Routing in Mobile Ad-hoc Networks (CORMAN). CORMAN includes three following important components. First, a new light-weight proactive source routing scheme *PSR* is proposed to provide source routing information in MANETs for both opportunistic data forwarding

and traditional IP forwarding. Second, we analyze and evaluate the topology change with mathematical model, and propose *large-scale live update* to update routing information more quickly with no extra communication overhead. Third, we propose the *small-scale retransmission* to utilize the broadcast nature one step further than ExOR, and furthermore it helps us to enhance the efficiency and robustness of the opportunistic data forwarding in MANETs. We run computer simulations in Network Simulator 2 (ns-2), and the simulation results indicate that the proposed solutions work well to support opportunistic data forwarding in MANETs. In particular, the routing overhead in PSR is only a small fraction of that in OLSR (Optimized Link State Routing), DSDV (Destination-Sequenced Distance Vector), and DSR (Dynamic Source Routing). Meanwhile, PSR has higher TCP (Transmission Control Protocol) throughput, much shorter packet end-to-end delay and delay variance than that in the three baseline protocols. Furthermore, a particular evaluation for the small-scale retransmission indicates us such a retransmission scheme can provide us up to 15% gains on the Packet Delivery Ratio (PDR) with UDP (User Datagram Protocol) data flows. At last, when we compare CORMAN as a system to AODV, we find the PDR in CORMAN is up to 4 times of the PDR in AODV.

Acknowledgements

Foremost, I would like to appreciate my supervisors sincerely, Dr. Yuanzhu Chen and Dr. Cheng Li, for their continuous support, guidance, and insightful advice. This thesis would not have been possible without the generous resources provided by the Faculty of Engineering & Applied Science and the Department of Computer Science with Memorial University. My research is supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada (Discovery Grants 293264-07 and 327667-2010, and Strategic Project Grant STPGP 397491-10), together with the fellowship from the School of Graduate Studies of Memorial University.

Then, I want to thank my wife Lan Wei. Lan accompanied me during the most important period of my research and looked after me. I really appreciate her understanding and support as I can not spend much time with her when I was doing research.

Last but not least, my acknowledgement and love also go to my parents, Zhiwei Wang and Qi Han, who have brought me to this wonderful world, taught me to appreciate everything I have, directed me through ups and downs, and stood by me at all time. I must ensure all these parties that I have had a tremendously rewarding experience and unforgettable time at Memorial University.

Table of Contents

List of Abbreviations

ACK	Acknowledgement
AODV	Ad hoc On-Demand Distance Vector Routing
BFST	Breadth First Spanning Tree
CBR	Constant Bit Rate
CCTS	Conservative Concurrent Transmitter Sets
CORMAN	Cooperative Opportunistic Routing in Mobile Ad-hoc Network
CTS	Concurrent Transmitter Sets
DSDV	Destination-Sequenced Distance Vector
DSR	Dynamic Source Routing
DV	Distance Vector algorithm
EAX	Expected Any-path Transmissions
ETX	Expected Transmission Count
ExOR	Extreme Opportunistic Routing
FC	Feasibility Condition
FO	Forwarding Order
FOA	Forwarding Order Acknowledgement
FTP	File Transfer Protocol
GCTS	Greedy Concurrent Transmitter Sets
IPv4	Internet Protocol version 4

LPA	Loop-free Path finding Algorithm
LS	Link State algorithm
MANET	Mobile Ad-hoc Network
MORE	MAC-independent opportunistic routing protocol
OLSR	Optimized Link State Routing
PDR	Packet Delivery Ratio
PFA	Path Finding Algorithm
PSR	Proactive Source Routing
RREP	Routing Reply
RREQ	Routing Request
RSSI	Received Signal Strength Indicator
SDF	Selection Diversity Forwarding
STAR	Source-Tree Adaptive Routing
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VANET	Vehicular Ad-hoc Network
VoIP	Voice over Internet Protocol
WRP	Wireless Routing Protocol
XL	Approximate Link state

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Introduction

A mobile ad hoc network is a wireless communication network, where nodes that are not within direct transmission range of each other will require other nodes to forward data. It can operate without existing infrastructure, supports mobile users, and falls under the general scope of multi-hop wireless networking. Such a networking paradigm was originated from the needs in battlefield communications, emergence operations, search and rescue, and disaster relief operations. Later, it found civilian applications such as community networks. A great deal of research results have been published since its early days in the 1980's [1]. The most salient research challenges in this area include end-to-end data transfer, link access control, security, and providing support for real-time multimedia streaming.

The network layer has received the most attention when working on mobile ad hoc networks. As a result, abundant routing protocols in such a network with differing objectives and for various specific needs have been proposed [2]. In fact, the two most important operations at the network layer, *i.e.*, data forwarding and routing, are distinct concepts. Data forwarding regulates how packets are taken from one link and put on another. Routing determines what path a data packet should follow from the source node to the destination. The latter essentially provides the former with control input.

Routing protocols in mobile ad hoc networks can be categorized using an array of criteria. The most fundamental difference among these is the timing of routing information exchange. On one hand, a protocol may require that nodes in the network should maintain valid routes to all destinations all the time. In this case, the protocol is considered to be *proactive*, a.k.a. *table driven*. Examples of proactive routing protocols include Destination-Sequenced Distance Vector (DSDV) [3] and Optimized Link State Routing (OLSR) [4]. On the other hand, if nodes in the network do not always maintain routing information, when a node receives data from the upper layer for a given destination, it must first find out how to reach the destination. This approach is called *reactive*, a.k.a. *on demand*. Dynamic Source Routing (DSR) [5] and Ad hoc On Demand Distance Vector (AODV) [6] fall in this category.

Even though a great deal of efforts in routing in ad hoc networks, data forwarding, in contrast, follows pretty much the same paradigm as in IP forwarding in the Internet. IP forwarding was originally designed for multi-hop wired networks, where one packet transmission can only be received by nodes attached to the same cable. For the case of modern Ethernet, an IP packet is transmitted at one end of the Ethernet cable and received at the other. However, in wireless networks, when a packet is transmitted over a physical channel, it can be detected by all other nodes within the transmission range on that channel. For the most part of the research history, overhearing a packet not intended for the receiving node had been considered as completely negative, *i.e.*, interference. Thus, the goal of research in wireless networking was to make wireless links as good as wired ones. Unfortunately, this ignores the inherent nature of broadcasting of wireless communication links. For mobile ad hoc networks to truly succeed beyond labs and testbeds, we must tame and utilize its broadcasting nature rather than fighting it. Cooperative communication is an effective approach to achieve such a goal.

The concept of cooperative communication was initially put forward as by Cover and El Gamal [7] studying the information theoretic properties of relay channels. More recent progress on this subject started to proliferate in the early 2000's [8]. In cooperative communication at the physical layer, multiple nodes overhearing the same packet may transmit it together as a virtual multiple-antenna transmitter. With enhanced digital signal-processing capabilities on the receiver side, the packet is more likely to be decoded. Yet, research on cooperative commutation at the link layer and above had been little until ExOR (Extreme Opportunistic Routing) [9]. ExOR is a milestone piece of work in this area and it is an elegant way to utilize the broadcasting nature of wireless links to achieve cooperative communication at the link and network layers of static multi-hop wireless networks. Therefore, in our research, we further extend the scenarios that the idea behind ExOR can be used, dubbed as *Cooperative Opportunistic Routing in Mobile Ad hoc Networks (CORMAN)*. Contributions of the systematic solution proposed in this thesis are highlighted as follows.

- We have designed a lightweight proactive source routing protocol so that each node has complete knowledge of how to route data to all other nodes in the network at any time. It has the same communication overhead as distance vector algorithms but provides each node with much more information about the network structure.

- When a flow of data packets are forwarded towards their destination, the route information carried by them can be adjusted by intermediate forwarders. Furthermore, as these packets are forwarded along the new route, such updated information is propagated upstream rapidly without any additional overhead. As a result, all upstream nodes learn about the new route at a rate much faster than via periodic route exchanges.

- We take opportunistic data forward to another level by allowing nodes that are not listed as intermediate forwarders to retransmit data if they believe certain packets are missing.

1.2 Thesis Organization

We organized the thesis as follows. We will present the related work done by other people in Chapter II. Furthermore, based on the related work review, we will highlight the motivations of our research and present the basic idea and the framework in our system. In particular, there are three important components in the system, including proactive source routing (PSR), large-scale live update, and small-scale retransmission. We will present the details of the three components mentioned above separately in Chapter III, Chapter IV, and Chapter V separately. We test the performance of the proposed schemes by using the Network Simulator 2 (ns-2) and the performance evaluation will be presented in Chapter VI. We not only evaluate the performance of CORMAN as an entire system but also evaluate the effectiveness of PSR and small-scale retransmission in particular. The thesis is concluded in Chapter VII with discussions and an overview of the future work.

Chapter 2

Related Work and Motivation

In this chapter, we review recent work in two related fields. We first review the opportunistic data forwarding, including its ancestors and derivations, the related math models built for it, and its effectiveness studies. We then review the importance of routing protocols, and hence we will review some routing protocols in the second part of this chapter. After the review of related work we will highlight the motivation and introduce the framework in our research.

2.1 Recent Work Related to Opportunistic Data Forwarding

This part provides an overview of opportunistic data forwarding in multi-hop wireless networks by explaining and comparing existed ones. They may include many trails for different coordinate protocols, metrics, network types and so on. The discussion in this part is organized as follows, Section 2.1.1 will present the mechanisms used to realize opportunistic data forwarding, and make comparison between these mechanisms. Section 2.1.2 will list other derivations in opportunistic data forwarding. Section 2.1.3

will give a review on the question that when the opportunistic data forwarding can get better performance than traditional IP forwarding in multi-hop network. Section 2.1.4 contains the mathematical models can help us to analyze the performance of opportunistic data forwarding and how to maximize such performances.

2.1.1 Protocols Based on Opportunistic Data Forwarding

In recent years, dozens of opportunistic data forwarding protocols have been proposed. Many of them are significant in this area. In this section, we present the challenges in opportunistic data forwarding and how to address them. First, we need to know where the advantages of opportunistic data forwarding comes from. Two most common scenarios that exist in wireless data transmission are shown in Figure 2.1 and Figure 2.2 [9].

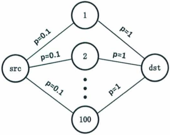


Figure 2.1: Topology and deliver probabilities as an example network

In Figure 2.1, the source node *src* transmits a packet to the destination node *dst*. By the topology and link delivery probabilities shown in the figure, we know the probability that at least one intermediate node receives the data from *src* is $1 - (1-0.1)^{100}$ which is greater than 10%, the probability of a particular one receives it.

Furthermore, any intermediate node can go on forwarding the packet. It is one reason why a better performance may be achieved in opportunistic data forwarding.

Figure 2.2: Best node can be used in opportunistic data forwarding

Figure 2.2 presents another possibility that one transmission may reach a node which is closer to the destination than the particular next hop in traditional IP forwarding. Assume in traditional IP forwarding, next hop for a packet sent from A is B, and node C overhears the packet by opportunity. In traditional approach, this is taken as the interference on node C and the overheard packet will be discarded. Meanwhile, node B will transmit it to C again. What makes it worse is that if node B can not decode the packet successfully and even though node C could properly decode the packet, the benefit can not be utilized in IP forwarding scheme, and the packet should be re-transmitted again by node A until it is received by B. Opportunistic data forwarding can postpone the decision to choose the best forwarder which may be far away from the transmitter but receives the packet by a certain opportunity. Note that, some situations may diminish the benefits of opportunistic data forwarding because the next hop is not determined and any node received the data could be an actual relay. Hence the duplicated transmission from many intermediate nodes should be avoided. If the cooperation protocol cannot guarantee that only one forwarder candidate will forward the data, duplicated transmission would decrease the performance. Moreover, if the coordination mechanism is quite resource consuming, this protocol will be unusable. Therefore, we believe that the coordination protocol, which serves opportunistic data forwarding, is the key point. How to design a coordination protocol for forward candidates will be a dominant issue to make opportunistic data forwarding more efficient and more practical. As far as we know, two protocols for coordination introduced below are significant for opportunistic data forwarding.

2.1.1.1 Multiple Handshake

Larsson [10] proposes a four-way handshake approach as the coordination protocol in his Selection Diversity Forwarding (SDF). In SDF, if a node has a packet to transmit, it just broadcasts the packet to every neighbor. Then, every neighbor received the packet successfully will send back an Acknowledgement (ACK) with their local information to the transmitter. The transmitter makes a decision based on the ACKs and sends a Forwarding Order (FO) to the best forwarder candidate. Once the selected relay node receives the FO, it will send the Forwarding Order ACK (FOA) back to the transmitter and then proceed to data forwarding. This process continues until the final destination is reached. As a piece of pioneering work on opportunistic data forwarding, Larsson has made a significant contribution. However, people realize two problems exist in Larsson's work. One is that the ACKs and FOA may be lost in the wireless environment, and the loss of either one of them will lead to unnecessary retransmissions. The other is that such a gossiping mechanism wastes a great deal of resources and introduce more delay. By this consideration, Rozner *et al.* [11] explored the approaches to make multiple-handshake coordination can tolerate ACK loss and reduce the delay. Rozner *et al.* uses selective ACKs to address ACK losses and minimize useless retransmissions, and use piggybacked ACKs and ACK compression to reduce ACK overhead. With selective ACKs, a single ACK loss will not trigger retransmissions because subsequent ACKs for later packets will guarantee that all packets have been received recently. In particular here, piggybacked ACKs are constructed by including acknowledgement information to a data frame, and when the packet is transmitted. The downstream nodes and upstream transmitter should decode the data and ACK information from such a packet separately. Another solution based on multiple-handshake is presented in Kurth *et al.* [12]. It suggests that a lower data rate can be used to transmit ACKs to enhance their reliability, and it also explores the transmitter diversity, which follows the same idea as receiver diversity. Naghshvar and Javidi [13] also use multiple-handshake to coordinate all forwarder candidates. They propose a new metric, which is the backlogs from all downstream links, to achieve a better performance. However, Naghshvar and Javidi assume that the ACKs are loss free.

2.1.1.2 Route-Prioritized Contention

Route-prioritized contention is the other type of coordination protocols. In route-prioritized contention, all forwarding candidates will follow a given order to content the wireless medium, and the *best* node that receives the packet should grab the medium first. Therefore, the broadcast nature can be explored.

This approach, as far as we know, is proposed by Biswas and Morris [9], called ExOR. ExOR is a milestone for the opportunistic data forwarding because of the following three reasons. 1) ExOR uses piggybacked ACK to tell the lower priority nodes that the packet has been forwarded by higher priority nodes. 2) The overhead in ExOR is small which only contains the ExOR header and the metric updating information. 3) Although it can not completely avoid duplications, it can avoid it in a high degree.

Now, we try to explain this coordinate protocol in detail. The header of ExOR is shown in Figure 2.3. In the figure, *Ver*, *HdrLen*, *PayloadLen*, *Checksum* and *Payload* are self-explanatory. In particular, to enhance the efficiency, ExOR delivers packets by *batch*, and each batch contains numbers of packets. The quantity of batches is recorded by the *BatchSz* in the header, so the *BatchSz* of every packet in the same batch should be identical. The data transmission is based on batch, which means the

Figure 2.3: ExOR header

source will flush all packets in the same batch into the network. Once all packets in the same batch are received by the destination (in ExOR, when the destination received over 90% of the total packets in the same batch), next batch should be transmitted. The *Batch ID* in the header identifies which batch the current packet belongs to, and the *PktNum* is the index of the packet in the current batch. *Forwarder List* contains the forwarder candidates which are selected and sorted by source node when a batch is constructed. As a result, the *Forwarder List* should keep the same for all packets in one batch. If a node contained in the forwarder list overhears a packet and the node's priority is higher than the transmitter's, the packet should be buffered in the node. In the forwarding step, to make a higher priority node send buffered packets first, waiting time is calculated by every node based on the its position in the *Forwarder List*. The time may be updated by the *Transmission Tracker*, which records the transmit rate and the quantity of packets that need transmit from current transmitter. To support the calculation, the *FragNum* and *FragSz* are added to the header. Here, a *Fragment* is a series packets heard by the node with ID *ForwarderNum*. *FragSz* is the quantity of packets that the current node should transmit when its turn comes, and *FragNum* is the index of this packet in current *Fragment*. When a node receives a packets for first time, the *Batch Map* maintained by the node should be updated. For every packet, the highest priority forwarder that has received it should be recorded in the *Batch Map*. A copy of this *Batch Map*, which is maintained on the node, will be added to the header of the packet to be transmitted. Hence, when lower priority nodes overhear a packet from a higher priority node, they will continue to merge the *Batch Map* maintained by themselves. At the same time, they will look up the *Batch ID* and *PktNum* in their buffers, and once a same packet is found, the transmission of this packet would be canceled. By looking up the *Batch Map*, many duplications can be avoided. All necessary information for coordinating forwarder candidates is integrated in the ExOR header, thus no time is wasted for gossiping between nodes. Many papers are published after ExOR, such as Zeng *et al.* [14], Yang *et al.* [15], Zhong *et al.* [16], and they use nearly the same coordinate protocol to make trails on other directions. However, ExOR has its disadvantage. Because the forwarding timer is always initialized according to the node's priority in the *Forwarder List*, the nodes far from destination will always wait for a long time. It quite constrains us from exploring the spatial reuse in the multi-hop network. Furthermore, ExOR is quite suitable for unicast, but in multicast, it may not perform well as observed by Chachulski *et al.* [17]. At last, piggybacked ACKs may be lost, so duplicated transmission may happen.

It is worthy to mention that Chachulski *et al.* [17] proposes MORE (Mac-independent opportunistic routing) which is an opportunistic forwarding approach with spatial reuse, and it also belongs to Route-Prioritized Contention scope. As far as we know, it is the first paper that uses network coding to realize opportunistic data forwarding. Its motivation of using network coding is that it wants to explore the spatial reuse in

ExOR and without duplicated transmission. Recall that, in ExOR the *Forwarder List* is generated by the source node, and the nodes which are far from the destination will wait for a long time, even though some of these nodes can transmit (or receive) some packets to (or from) some other nodes without much interference on the forwarding taking place far away. The reason why ExOR cannot operate like a pipeline is that for each packet we cannot tell how far it has already been transmitted in last hop. To enable pipeline opportunistic data forwarding without unnecessary data transmission, the network coding is used by Chachulski.

In MORE, some predefined number of packets compose a batch as that in ExOR. All packets in the same batch will be encoded by linear network coding before sending out from source node. The source node will calculate a sorted *Forwarder List* by the routing metric of expected transmission count (ETX) [18], which is also quite similar to ExOR. When a node in the *Forwarder List* receives a packet of a batch, and if this packet is linearly independent from all received packets in the same batch, the packet should be forwarded. The exact time to forward the packet depends on two factors: one is the position of the forwarder in the *Forwarder List*, and the other factor depends on the 802.11 MAC. The difference from ExOR is the time used to wait for transmission can be shortened, where if the timer expires and the medium is free, the linear encoded packet can be transmitted.

2.1.2 Other Variants of Opportunistic Forwarding

Besides the contributions on coordination protocol for opportunistic data forwarding, several work makes many other trials by changing the network types, metrics, or combining with other technologies to help opportunistic data forwarding make decisions.

2.1.2.1 Network Types

Multi-hop wireless networks is like a family which is composed by mesh networks, mobile ad-hoc networks, sensor networks, and vehicular networks. Each of them can borrow the idea of opportunistic data forwarding from mesh networks to promote theirs performance. By following this way, Ma *et al.* [19] uses opportunistic forwarding in sensor networks to enhance the probability that a data packet is forwarded successfully. Hence, more sensor nodes can keep in the sleep mode for a longer time to save energy without influencing the performance. Vehicular networks can also explore the benefits of opportunistic data forwarding, and usually they use the positioning service to get the topology of the vehicular networks and select the *best* forwarder by the perspective of distance, like in Leontiadis and Mascolo [20].

2.1.2.2 Metrics

In ExOR, ETX is used as the metric to evaluate which candidate is better. However, numbers of metrics have been proposed recently, such as distance, expected any-path transmissions, and backlog. Distance is quite suitable for position-based opportunistic forwarding because it is quite easy to be calculated by positioning service and has been shown to have relatively good performance in Leontiadis and Mascolo [20], Yang *et al.* [15]. Expected any-path transmissions (EAX) is proposed in Zhong *et al.* [16], which equals to the expected number of transmissions required to deliver a packet to its destination under the perspective of opportunistic forwarding.

2.1.2.3 Network Coding Based Opportunistic Forwarding

Network Coding based opportunistic data forwarding has been mentioned in the previous section, which usually uses the linear network coding method to encode the received or original packets in a series of random linear combinations. The exploration on this includes Chachulski *et al.* [17], Radunović *et al.* [21], and Koutsonikolas *et al.* [22]. The coordination protocols will not work so aggressively in ExOR. Usually ACKs should be sent back when the destination receives enough information to decode all packet in the batch, and the source node will stop sending packets or go on sending the packets in next batch when the ACK for current batch is received.

2.1.2.4 Position Based Opportunistic Forwarding

Position based opportunistic forwarding is explored in Yang *et al.* [15]. The metric is the distance from any forward candidates to the final destination, and the coordinate protocol is quite similar with that in ExOR. The node which is closer to the destination will access the medium earlier to forward buffered packets.

2.1.3 Scenarios Suitable for Opportunistic Forwarding

In the previous sections, we have presented the advantages of opportunistic data forwarding and the challenges in it. In work done by Shah *et al.* [23] and Kim *et al.* [24], they make a comparison between traditional IP forwarding and opportunistic data forwarding. They believe the opportunistic data forwarding is suitable for the networks whose mobility is not high, and the higher the node density, the better the performance that can be achieved. The reasons to explain this is that, if the mobility is high, the forwarder candidates cannot always keep in communication range with each other, so a packet may be transmitted by two or more forwarders for several times. The duplication becomes the most important reason that decreases the benefits we get from opportunistic data forwarding, the performance gain by opportunistic forwarding will be marginal or even negative because the coordination overhead always exists in coordination protocols.

2.1.4 Performance Modeling for Opportunistic Forwarding

Many performance analysis and optimization work made their contributions in the study of opportunistic data forwarding as well. These work is categorized in two directions. One is building models to analyze opportunistic data forwarding, and the other is using existing models to enhance the performance in opportunistic data forwarding.

2.1.4.1 Performance Optimization

No matter what kind of data transmission protocols people choose to use, they want to optimize the network performance. Moreover, if a boundary of performance can be proved, it will be quite useful. Radunović *et al.* [21] and Zeng *et al.* [25] make their contributions on the performance optimizations. In opportunistic data forwarding, many forwarder candidates can overhear the same packet, but different forwarding decision will affect the performance significantly. Hence, Radunović *et al.* uses an optimal flow-decision to maximize the links' utility of the whole network. A comprehensive model built by Radunović *et al.* gives us the relationship between the links' utility and flow-decision-set. To maximize the utility, Radunović *et al.* handle the optimization problem with two steps. First, the transport credits is proposed and it is used to denote the quantity of packets that have been sent out. Second, for each possible flow on each node, Radunović studies the relationship between the flow's transport credits and a three-tuple composed by *scheduling strategy, power control* and *transmit rate control*. Hence, the transport credits for a particular node can be presented by the three-tuple, so the summation of all nodes' transport credits in the network can be related with the three-tuple as well. When we take the three-tuple as the varying variable, the optimized network utility would be achieved by selecting the tuples that give us the maximum value of transport credits.

In the work by Zeng *et al.*, they use "*Transmitter Conflict Graph*" rather than "*Link Conflict Graph*" to find what are the lower and upper bounds throughput in the opportunistic forwarding based multi-hop wireless networks. Three important concepts are introduced in Zeng *et al.* [25]. They are Concurrent Transmitter Sets (CTS), Conservative CTS (CCTS), and Greedy CTS (GCTS). CTS is self-explained, and CCTS is one extreme case in CTS that all links associated with any node in the CCTS can be used simultaneously. GCTS is another case in CTS that at least one link of any node in the node set can be used concurrently. Hence, the lower bound and the upper bound can be calculated in GCTS and CCTS separately. The lower bound can be calculated in protocol model with linear programming and the upper bound can be figured out in the physical model because of the fact that SNR will decrease when more concurrent links are used.

2.1.4.2 Modeling from Markov Process and Game Theory

[26] done by Cerdà-Alabern *et al.* and [27] done by Lu *et al.* are two typical work based on the Markov Process. The idea relies on the packet received probability between every node pair in a network can be estimated, either by radio's propagation model or by exchanging the probe messages periodically. Moreover, in opportunistic forwarding, the node with the smallest metric should forward the data first, so we can statistically predict the performance for different metrics, such as hop count, transmission count and even energy consumption. In Lu *et al.* [27], the Markov Process is constructed with the states which are the combinations of different events, and the probability distribution of the ETX for a node to deliver a packet is proposed. However, in Cerdà-Alabern *et al.* [26], they just use the forwarder candidates to be the states in Markov Process. The receive probability between every pair of nodes is estimated by radio propagation model. Therefore, the expectations of hop count and

throughput could be found.

Game Theory is used in Wu *et al.* [28], and the purpose of their work is to try to build a mechanism by which the rewards and punishments are given to every node to make them be honest to announce its link quality and help other nodes to forward packets. The reason to do such work is that nodes in opportunistic forwarding may be selfish, and such a behavior is unfair for other node and degrades the total performance. In Wu *et al.* [28], they use Strict Dominant Strategy Equilibrium to design a mechanism in which nodes will get theirs benefit most if and only if they keep honest to report link quality and do their best to forward packets for others.

2.2 Routing Algorithms Review for Opportunistic Data Forwarding in MANETs

In this section, we will briefly review the existing routing protocols. The routing protocol is important for our CORMAN because we want to utilize the Route-Prioritized Contention (Section 2.1.1.2) approach to equip Mobile Ad-hoc Networks (MANETs) the ability of opportunistic data forwarding. This thesis aims to present the CORMAN, a series of solutions of opportunistic data forwarding in MANETs, so we need to investigate that whether the existing routing protocols could support opportunistic data forwarding and how they perform if they could.

2.2.1 Timing Strategy in Routing Protocols

Basically, routing protocols in mobile ad hoc networks can be categorized using an array of criteria. The most fundamental among these is the timing of routing information exchange. On one hand, a protocol may require that nodes in the network should maintain valid routes to all destinations all the time. In this case, the protocol is considered to be *proactive*, a.k.a. *table driven*. Examples of proactive routing protocols include Destination-Sequenced Distance Vector (DSDV) [3] and Optimized Link State Routing (OLSR) [4]. On the other hand, if nodes in the network do not always maintain routing information, when a node receives data from the upper layer for a given destination, it must first find out how to reach the destination. This approach is called *reactive*, a.k.a. *on demand*. Dynamic Source Routing (DSR) [5] and Ad hoc On Demand Distance Vector (AODV) [6] fall in this category.

2.2.2 Two Basic Algorithms — Link State and Distance Vector

These well-known routing schemes can also be categorized by their fundamental algorithms. The most important types of algorithms in routing protocols are Distance Vector (DV) and Link State (LS) algorithms. In LS, every node will share its best knowledge of links with other nodes in the network, so nodes can reconstruct the topology of the entire network locally, *e.g.*, OLSR. In DV, a node only provides its neighbors the cost to every given destination; so nodes know the costs to a given destination via different neighbors as the next hop, *e.g.*, DSDV and AODV. DSR is an early source routing protocol in MANETs. In DSR, when a node has data for a destination node, a route request is flooded to all nodes in the network. An intermediate node which has received the request adds itself in the request and rebroadcasts it. When the destination node receives the request, it transmits a route reply packet to send the discovered route back to the source node via the route in request packet reversely, and the source node will use the discovered route in data transmission.

2.2.3 Tree Based Routing Protocols Derived from the Internet

As the scalability of the Internet also suffers from the overhead problem, many lightweight routing protocols have been proposed for the Internet. In the pioneering work, which is done by Garcia-Luna-Aceves and Murthy [29], a new routing algorithm called Loop-free Path-finding Algorithm (LPA) is proposed based on path-finding algorithm (PFA). To avoid loops, LPA proposed a Feasibility Condition (FC). The FC can effectively avoid the temporary loop but it may overkill some possible valid path in the network. Another piece of work, which is done by Behrens and Garcia-Luna-Aceves [30], a new routing algorithm called Link Vector (LV) algorithm is proposed. In LV, the basic element in the update message contains the destination, the precursor to destination, the cost between them, and a sequence number to avoid loops. LV is different from LS, because in LS, an updater sends all the link information it knows by flooding, but in LSA, an updater only send the information of links which are preferred to use. So the node in LV can only construct a partial topology of the entire network. A recent work done by Levchenko *et al.* [31] called *approximate link state* (XL) conclusively summarize the LV and PFA in a nutshell. It uses *soundness* and *completeness* to define the correctness of the routing protocols and uses *stretch* to evaluate the optimality. XL also uses *lazy* update concept to further decrease the overhead. So, in XL, node sends update message that only contains the changed links which are on the only way that must be passed to a given destination, or the updated link can improve the cost to a destination node by a given parameter.

Similar attempts to reduce wireless routing overhead have been made during the development listed above. Murthy and Garcia-Luna-Aceves proposed Wireless Routing Protocol (WRP) [32], which utilizes the basic idea of PFA within wireless networks.

Every node in WRP has a tree structure for the network, and every time the node only send out the routing update message by differential update. However, WRP requires the receiver of an update message transmit the ACK. Such requirment introduces more overhead, consumes more channel resources, but it may not improve the performance dramatically. Furthermore, WRP uses *RouterDeadInterval* to detect the loss of neighbors which may adversely affect the response time of topology change, considering only one route is maintained in the tree structure. An extensional method based on WRP is Source Tree Adaptive Routing (STAR) [33], which is proposed by Garcia-Luna-Aceves and Spohn. In STAR, every node maintains a tree structure for the network, and adopts a tree update strategy that is neither proactive nor reactive. Instead, it uses a lazy approach, where update message will only be transmitted when the local tree structure is considered sufficiently inferior to the original optimum.

2.2.4 Suitability of Existing Routing Protocols for Opportunistic Data Forwarding in MANETs

In fact, none of previous listed protocols can ideally support opportunistic data forwarding in MANETs. In particular, AODV [6], DSDV [3], and other DV-based routing algorithms were not designed for source routing; so they may not suitable for opportunistic data forwarding. The reason is that every node in these protocols only knows the next hop to reach a given destination node but not the complete path. OLSR [4] and other LS-based routing protocols could support source routing but their overhead is still fairly high for the load-sensitive MANETs. DSR, together with its derivations, such as [34], [35] and [36], are not suitable because they have long bootstrap delay and are therefore not efficacious for frequent data exchange. Furthermore, the reactive routing protocols will inject too many route request packets in the mobile networks, especially when there are a large number of data sources. Moreover, the route reply message may be lost since it is sent based on IP forwarding via recorded route reversely, so reactive routing schemes suffer from even more unpredictable delay in data transmission. The WRP [32] and STAR [33], the early attempts to port the routing capabilities in link state routing protocols to MANETs, are built on the framework of PFA for each node to use a tree for loop-free routing. Although WRP is an innovative exploration in the research on MANETs and STAR uses "lazy" routing strategy to reduce topology update burden, they have a very high operation overhead due to the routing procedures are triggered by the event of topology changes, which introduce a great deal of information exchanged and stored by the nodes. Our intention was to include WRP in our experimental comparison later in this thesis, and we have implemented WRP in ns2. Unfortunately, our preliminary tests indicate that its communication overhead is at least an order of magnitude higher than the other main-stream protocols.

In a nutshell, design a new light-weight proactive source routing protocol is a very important component to develop opportunistic data forwarding in MANETs.

2.3 Motivation and Framework

When we review the related work we can see that a systematic solution to equip MANETs the ability of opportunistic data forwarding is required. Hence, in our research, we propose CORMAN (Cooperative Opportunistic Routing in Mobile Adhoc Networks) as a network layer solution to implement opportunistic data forwarding in MANETs. Its node coordination mechanism is largely in line with that of ExOR and it is an extension to ExOR in order to accommodate node mobility. Here, we first highlight our objectives and challenges to achieve them. Later in this section, we provide a general description of CORMAN. The details of the major components of CORMAN will be presented in Chapter III (proactive source routing), Chapter IV (large-scale live update), and Chapter V (small-scale retransmission).

2.3.1 Objectives and Challenges

CORMAN has two objectives: 1) to broaden the applicability of ExOR to mobile multi-hop wireless networks without relying on external information sources, such as node positions; 2) to incur a smaller overhead than ExOR by including shorter forwarder lists in data packets.

The following challenges are thus encountered.

1. Overhead in route calculation — CORMAN relies on the assumption that every source node has complete knowledge of how to forward data packets to any node in the network at any time. This calls for a proactive source routing protocol. Link state routing, such as OLSR, would meet our needs but it is fairly expensive in terms of communication costs. Therefore, we need a lightweight solution to reduce the overhead in route calculation.

2. Forwarder list adaptation — When the forward list is constructed and installed in a data packet, the source node has updated knowledge of the network structure within its proximity but its knowledge about further areas of the network can be obsolete due to node mobility. This becomes worse as the data packet is forwarded towards the destination node. To address this issue, intermediate nodes should have the ability to update the forwarder list adaptively with their new knowledge when forwarding data packets.

3. Robustness against link quality variation — When used in a dynamic environment, a mobile ad hoc network must inevitably face drastic link quality fluctuation. A short forwarder list carried by data packets implies that they tend to

take long and possibly weak links. For opportunistic data transfer, this could be problematic since the list may not contain enough redundancy in selecting intermediate nodes. This should be overcome with minimal additional overhead.

2.3.2 CORMAN Fundamentals

CORMAN forwards data in a similar batch-operated fashion as ExOR. A flow of data packets are divided into batches. All packets in the same batch carry the same forwarder list when they leave the source node. To support CORMAN, we have an underlying routing protocol, *Proactive Source Routing* (PSR), which provides each node the complete routing information to all other nodes in the network. Thus, the forwarder list contains the identities of the nodes on the path from the source node to the destination. As packets progress in the network, the nodes listed as forwarders can modify the forwarder list if any changes have been observed in the network topology. This is referred to as *large-scale live update* in our work. In addition, we also allow some other nodes that are not listed as forwarders to retransmit data if this turns out to be helpful, referred to as *small-scale retransmission*. Note that CORMAN is a completely network layer solution and can be built upon off-the-shelf IEEE 802.11 networking products without any modification.

Therefore, the design of CORMAN has the following three modules. Each module answers to one of the challenges stated previously.

1. Proactive source routing — PSR runs in the background so that nodes periodically exchange network structure information. It converges after the number of iterations equal to the network diameter. At this point, each node has a spanning tree of the network indicate the shortest paths to all other nodes. The amount of information broadcast by each node in an iteration is $O(n)$, where n is the number of nodes in the network. Such an overhead is the same as

distance vector algorithms and much smaller than that in link state algorithms. Technically, PSR can be used without CORMAN to support conventional IP forwarding.

2. Large-scale live update — When data packets are received by and stored at a forwarding node, the node may have a different view of how to forward them to the destination from the forwarder list carried by the packets. Since this node is closer to the destination than the source node, such discrepancy usually means that the forwarding node has more updated routing information. In this case, the forwarding node updates the part of the forwarder list in the packets from this point on towards the destination according to its own knowledge. When the packets with this updated forwarder list are broadcast by the forwarder, the update of network topology change propagates back to the upstream neighbor. The neighbor incorporates the change to the packets in its cache. When these cached packets are broadcast later, the update is further propagated towards the source node. Such an update procedure is significantly faster than the rate at which a proactive routing protocol disseminates routing information.

3. Small-scale retransmission — A short forwarder list forces packets to be forwarded over long, and possibly weak, links. To increase the reliability of data forwarding between two listed forwarders, CORMAN allows nodes that are not on the forwarder list but are situated between these two listed forwarders to retransmit data packets if the downstream forwarder has not received these packets successfully. Since there may be multiple such nodes between a given pair of listed forwarders, CORMAN coordinates retransmission attempts among them extremely efficiently.

2.4 Summary

In this chapter, we first review the related work in two fields: one is how to utilize the broadcast nature in wireless networks, including the protocols proposed by other people, important performance modelings, and the analysis of the effectiveness of opportunistic data forwarding; the other field is about the routing schemes, and we review them within proactive and reactive categories and introduce the three basic concepts behind them — the link state, distance vector, and tree based routing. Based on the review work, we highlight the motivation of our research is to implement the opportunistic data forwarding in MANETs, and to equip MANETs the ability to utilize broadcast nature in data transmission. In the final part of this chapter, we put forward the challenges in the system and propose the basic ideas of our solutions.

Chapter 3

Proactive Source Routing — PSR

Before describing the details of PSR, we will first review some graph theoretic terms used here. Let us model the network as an undirected graph $G = (V, E)$, where V is the set of nodes (or vertices) in the network and E is the set of wireless links (or edges). Two nodes u and v are connected by an edge $e = (u, v) \in E$ if they are close to each other and can communicate directly with a given reliability. Given a node v, we use $N(v)$ to denote its open neighborhood , *i.e.*, $\{u \in V | (u, v) \in E\}$. Similarly, we use $N[v]$ to denote its closed neighborhood, *i.e.*, $N(v) \cup \{v\}$. In general, for a node v, we use $N_l(v)$ $(l \geq 1)$ to denote the distance-l open neighborhood of v, *i.e.*, the set of nodes that are exactly l hops away from v. Similarly, $N_l[v]$ $(l \geq 1)$ denotes the distance-l closed neighborhood of v, *i.e.*, the set of nodes that are within l hops of v. As special cases, $N_1[v] = N[v]$, $N_1(v) = N(v)$, $N_0[v]$ is v itself, and $N_0(v) = \emptyset$. Also as a convention in graph theory, for any $S \subseteq V$, we use $\langle S \rangle$ to denote the subgraph induced by S, *i.e.*, $\langle S \rangle = (S, E')$, where E' is the set of edges where each element

The research findings from CORMAN entirely have been published to IEEE Journal in Selected Areas in Communications [37]

The research findings from PSR and its early refinement have been published to IEEE GLOBECOM'11 [38] and IEEE Communications Letters [39]

PSR's last improvement has been submitted to IEEE INFOCOM'12 [40] has both endpoints in S. The readers are suggested to refer to the monograph of West [41] for other graph theoretic notions and other details.

3.1 Design of PSR

Essentially, PSR provides every node with a Breadth First Spanning Tree (BFST) of the entire network rooted at itself. To do that, nodes periodically broadcast the tree structure to its best knowledge in each iteration. Based on the information collected from neighbors during the most recent iteration, a node can expand and refresh its knowledge about the network topology by constructing a deeper and more recent BFST. This knowledge will be distributed to its neighbors in the next round of operation (Section 3.1.1). On the other hand, when a neighbor is deemed lost, a procedure is triggered to remove its relevant information from the topology repository maintained by the detecting node (Section 3.1.2). Intuitively, PSR has about the same communication overhead as distance-vector-based protocols. We go an extra mile to reduce the communication overhead incurred by PSR's routing agent. Details about such overhead reduction will be discussed in Section 3.1.3.

3.1.1 Route Update

Due to its proactive nature, the update operation of PSR is iterative and distributed among all nodes in the network. At the beginning, a node v is only aware of the existence of itself, so there is only a single node in its BFST, which is the root node v. By exchanging the BFSTs with the neighbors, it is able to construct a BFST within $N[v]$, *i.e.*, the star graph centered at v, denoted by S_v.

In each subsequent iteration, nodes exchange their spanning trees with their neighbors. From the perspective of node v, towards the end of each operation interval, it has received a set of routing messages from its neighbors packaging the BFSTs. Note that, in fact, more nodes may be situated within the transmission range of v, but their periodic updates were not received by v due to, say, bad channel conditions. After all, the definition of a neighbor in MANETs is a fickle one. (We have more details on how we handle lost neighbors subsequently.) Node v incorporates the most recent information from each neighbor to update its own BFST. It then broadcasts this tree to its neighbors at the end of the period. Formally, v has received the BFSTs from some of its neighbors. Including those from whom v has received updates in recent previous iterations, node v has a BFST, denoted T_u, cached for each neighbor $u \in N(v)$. Node v constructs a union graph

$$G_v = S_v \cup \bigcup_{u \in N(v)} (T_u - v)\,. \tag{3.1}$$

Here, we use $T - x$ to denote the operation of removing the subtree of T rooted at node x. As special cases, $T - x = T$ if x is not in T and $T - x = \emptyset$ if x is the root of T. Then, node v calculates a BFST of G_v, denoted T_v, and places T_v in a routing packet to broadcast to its neighbors.

In fact, in our implementation, the above update of the BFST happens multiple times within a single update interval so that a node can incorporate new route information to its knowledge base more quickly. To the extreme, T_v is modified every time a new tree is received from a neighbor. Apparently, this is a trade-off between the routing agent's adaptivity to network changes and computational cost. Here, we choose routing adaptivity as a higher priority assuming that the nodes are becoming increasingly powerful in packet processing. Nevertheless, this does not increase the communication overhead at all because one routing message is always sent per update interval.

Assume that the network diameter, *i.e.*, the maximum pairwise distance, is D hops. After D iterations of operation, each node in the network has constructed a BFST of the entire network rooted at itself. This information can be used for any source routing protocol. The amount of information that each node broadcasts in an iteration is bounded by $O(|V|)$ and the algorithm converges in at most D iterations.

3.1.2 Neighborhood Trimming

The periodically broadcast routing messages in PSR also double as "Hello" messages for a node to identify which other nodes are its neighbors. When a neighbor is deemed lost, its contribution to the network connectivity should be removed, called "neighbor trimming". Consider node v, the neighbor trimming procedure is triggered at v about neighbor u when

- no routing update or data packet has been received from this neighbor for a given period of time, or
- a data transmission to node u has failed as reported by the link layer.

Node v responds by

1. first updating $N(v)$ with $N(v) - \{u\}$,
2. next constructing the union graph with the information of u removed

$$G_v = S_v \cup \bigcup_{w \in N(v)} (T_w - v)\,, \tag{3.2}$$

3. and then computing the BFST T_v.

Notice that T_v thus calculated is not broadcast immediately to avoid excessive messaging. With this updated BFST at v, it is able to avoid sending data packets via lost neighbors. Thus, multiple such neighbor trimming procedures may be triggered within one period.

3.1.3 Streamlined Differential Update

In addition to dubbing route updates as hello messages in PSR, we interleave the "full dump" routing messages as stated previously with "differential updates". The basic idea is to send the full update messages less frequently than shorter messages containing the difference between the current and previous knowledge of a node's routing module. Both the benefit of such an approach and how to balance between these two types of messages have been studied extensively in earlier proactive routing protocols. In this work, we further streamline the routing update in two new avenues. First, we use a compact tree representation in full-dump and differential update messages to halve the size of these messages. Second, every node attempts to maintain a "stable" BFST as the network changes so that the differential update messages are even shorter.

- Compact tree representation — For the full dump messages, our goal is to broadcast the BFST information stored at a node to its neighbors in a short packet. To do that, we first convert the general rooted tree into a binary tree of the same size, say s nodes. Then we serialize the binary tree using a bit sequence of $34 \times s$ bits, where the IPv4 (Internet Protocol version 4) is assumed. Specifically, we scan the binary tree layer by layer. When processing a node, we first include its IP address in the sequence. In addition, we append two more bits to indicate if it has the left and/or right child. For example, the binary tree in Figure 3.1 is represented as `A10B11C11D10E00F00G11H00I00`. As such, the size of the update message is a bit over half compared to the traditional approach, where the message contains a discrete set of edges.

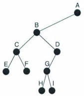


Figure 3.1: Binary Tree

The difference between two BFSTs can be represented by the set of nodes who have changed parents, which are essentially a set of edges connecting to the new parents. We observe that these edges are often clustered in groups. That is, many of them form a sizeable tree subgraph of the network. Similar to the case of full dump, rather than using a set of loose edges, we use a tree to package the edges connected to each other. As a result, a differential update message usually contains a few small trees, and its size is noticeably shorter.

- Stable BFST — The size of a differential update is determined by how many edges it includes. Since there can be a large number of BFSTs rooted at a given node of the same graph, we need to alter the BFST maintained by a node as little as possible when changes are detected. To do that, we modify the computation described earlier in this section such that a small portion of the tree needs to change either when a neighbor is lost or when it reports a new tree.

  Consider node v and its BFST T_v. When it receives an update from neighbor u, denoted by T_u, it first removes the subtree of T_v rooted at u. Then it incorporates the edges of T_u for a new BFST. Note that the BFST of $(T_v - u) \cup T_u$ may not contain all necessary edges for v to reach every other node. Therefore, we still need to construct the union graph

$$(T_v - u) \cup \bigcup_{w \in N(v)} (T_w - v)\,, \tag{3.3}$$

before calculating its BFST. To minimize the alteration of the tree, we add one edge of $(T_w - v)$ to $(T_v - u)$ at a time. During this process, when there is a tie, we always try to add edges that were originally removed from T_v.

When node v thinks that a neighbor u is lost, it deletes the edge (u, v) but still utilizes the network structure information contributed by u earlier. That is, even if it has moved away from v, node u may still be within the range of one of v's neighbors. As such, T_v should be updated to a BFST of

$$(T_v - u) \cup (T_u - v) \cup \bigcup_{w \in N(v)} (T_w - v)\,. \tag{3.4}$$

Note that, since $N(v)$ no longer contains u, we need to explicitly put it back into the equation. Similarly in this case, edges of $(T_u - v) \cup \bigcup_{w \in N(v)} (T_w - v)$ are added to $(T_v - u)$ one at a time, with those just removed because of u taking priority.

3.2 Implementation

As the implementation contains a great deal of algorithms, here we can only introduce some important ones. Before looking at the algorithms, we summarize the notations we used for these algorithms in Table 3.1

Table 3.1: Notations in algorithms

Notation	Explanation
$stack \Uparrow [v]$	Pop the top of *stack* and store to v, v is optional
$stack \Downarrow v$	Push v on the top of *stack*
$queue \Uparrow [v]$	Get an element from FIFO *queue* and keep in optional v
$queue \Downarrow v$	Put v into *queue* in FIFO manner
$\mathrm{R}(T)$	Get the root node of tree T
$\mathrm{B}_{\mathrm{l/r}}(n)$	The left/right brother node of n
$\mathrm{L}_{\mathrm{T}}(n)$	Look up node with same ID of n from tree T
$\mathrm{G}_{\mathrm{s/b/l}}(n)$	Create a spanning/bianry tree node or linear element of n
$\mathrm{A}_{\mathrm{T}}(n^p, n^c)$	Add a new node n^c as a child of n^p in T
$\mathrm{U}_{\mathrm{T}}(n^p, n^c)$	Update n^p as the new parent of n^c in tree T
$\mathrm{H}_{\mathrm{T}}(n)$	Hops from node n to its root node in tree T
$\mathrm{C}_{\mathrm{x}}(n)$	The x^{th} child of node n in spanning tree
$\mathrm{C}_{\mathrm{l/r}}(n)$	The left/right child of node n in binary tree
$\mathrm{P}(n)$	The parent of node n
$\mathrm{c}_{\mathrm{l}}(e)$	The left tag of a linear element e
$\mathrm{c}_{\mathrm{r}}(e)$	The right tag of a linear element e
$a \cup e$	append an element e at the end of an array a

3.2.1 Routing and Neighborhood Update Algorithm

As the operations in routing update and neighborhood update are similar, we take the routing update as an example to introduce their implementation. The Formula 3.3 theoretically provide us how to implement routing update algorithm. However, direct implementation needs high algorithm complexity, and what worse is that, according to the procedure in Formula 3.3, every time a node v receives an update from neighbor u_k, we have to check every node in subtrees $T_w - v$ $(w \in N(v))$. That is a time consuming job in both reality and simulation. In fact, from Section 3.1.1 we can see that the shorter path is always selected, and if two pathes with same hops are found in the union graph, we always keep the original one. Hence in our implementation, we look up every node in trees $T_w - v$ $(w \in N(v))$ from the BFST being constructed. If a duplicated node is found and the new discovered path is shorter, the parent of the node should be changed. Therefore, the incorporating work for every subtree

$T_w - v$ ($w \in N(v)$) can be finished by Algorithm 3.1.

Algorithm 3.1 Incorporate $\mathcal{T}_{u_i} - v$ ($u_i \in N(v)$) to $\mathcal{T}_v$

```
stack ⇓ R(T_{u_i} − v)
loop
  stack ⇑ n^c_{u_i}
  if n^c_{u_i} is null then
    if stack is not null then
      stack ⇑ n^c_{u_i}; stack ⇓ B_r(n^c_{u_i})
    else
      return;
    end if
  else
    n^c_v ⇐ L_{T_v}(n^c_{u_i})
    if stack is null then
      n^p_v ⇐ R(T_v)
    else
      stack ⇑ n^p_{u_i}; n^p_v ⇐ L_{T_v}(n^p_{u_i}); stack ⇓ n^p_{u_i}
    end if
    if n^c_v is null then
      n^c_v ⇐ G_s(n^c_{u_i}); A_{T_v}(n^p_v, n^c_v)
    else if H_{T_v}(n^c_v) > H_{T_{u_i}−v}(n^c_{u_i}) + 1 then
      U_{T_v}(n^p_v, n^c_v)
    end if
    stack ⇓ n^c_{u_i}
    if n^c_{u_i} is leaf node then
      stack ⇓ null
    else
      stack ⇓ C_0(n^c_{u_i})
    end if
  end if
end loop
```

The routing update for neighbor u_k would be finished if we run the Algorithm 3.1 for all subtrees in the last parameter of Formula 3.3. Hence the implementation of neighborhood update would be finished by the same way, and the only difference is the initial $\mathcal{T}_v$.

3.2.2 Algorithms for Transformation of Tree Structures

In general, each node maintains the topology by BFST as we introduced before, which is quite suitable to provide a route to a destination node. When a routing update packet is needed, either the entire spanning tree in a full dump update or the forest in a differential update should be presented in linear form. In fact, a single tree can be taken as a forest with one member. As we discussed before, we first convert a spanning forest to a binary forest, and then convert the binary forest into its linear form. When a node receives the update, receiver should restore the binary form of a linear expression and then transform the binary forest to spanning forest. Depending on which kind of update is in used, we can choose how to deal with the spanning forest. If the update is a full dump one, there must be only one tree in the spanning forest, and we direct replace the original BFST cached for such neighbor by the new one. Otherwise, we should update the original BFST according to the spanning forest as introduced in Section 3.2.3. In this part we only talk about the algorithms related with the transformation of tree structures. In a nutshell, we have five algorithms to help us finish all these work:

- Algorithm to convert spanning tree to binary tree — Algorithm 3.2.
- Algorithm to convert binary tree to spanning tree — Algorithm 3.3.
- Algorithm to convert binary tree to linear-tree — Algorithm 3.4.
- Algorithm to convert linear-tree to binary tree — Algorithm 3.5.
- Last one is the ancillary algorithm to separate the linear-forest to a set of linear-trees — Algorithm 3.6.

Algorithm 3.2 Convert spanning tree $\mathcal{T}_v$ to binary tree $\mathcal{T}_{v'}$

Initialize $\mathcal{T}_{v'}$ with node v as root
$stack \Downarrow \mathrm{R}(\mathcal{T}_v)$
$w^{\mathrm{c}}_{v'} \Leftarrow \mathrm{R}(\mathcal{T}_{v'})$
loop
    $stack \Uparrow n^{\mathrm{c}}_v$
    if n^{c}_v is null **then**
        if *stack* is not null **then**
            $stack \Uparrow n^{\mathrm{c}}_v$
            for $i = 1$ to $\mathrm{N_c}(n^{\mathrm{c}}_v)$ **do**
                $w^{\mathrm{c}}_{v'} \Leftarrow \mathrm{P}(w^{\mathrm{c}}_{v'})$
            end for
            $n^{\mathrm{c}}_v \Leftarrow \mathrm{B_r}(n^{\mathrm{c}}_v)$
            if n^{c}_v is not null **then**
                $n^{\mathrm{c}}_{v'} \Leftarrow \mathrm{G_b}(n^{\mathrm{c}}_v)$; $\mathrm{P}(n^{\mathrm{c}}_{v'}) \Leftarrow w^{\mathrm{c}}_{v'}$; $\mathrm{C_r}(w^{\mathrm{c}}_{v'}) \Leftarrow n^{\mathrm{c}}_{v'}$; $w^{\mathrm{c}}_{v'} \Leftarrow n^{\mathrm{c}}_{v'}$
            end if
            $stack \Downarrow n^{\mathrm{c}}_v$
        else
            return;
        end if
    else if n^{c}_v is leaf node **then**
        $stack \Downarrow$ null
    else
        $n^{\mathrm{c}}_v \Leftarrow \mathrm{C_0}(n^{\mathrm{c}}_v)$; $n^{\mathrm{c}}_{v'} \Leftarrow \mathrm{G_b}(n^{\mathrm{c}}_v)$; $\mathrm{P}(n^{\mathrm{c}}_{v'}) \Leftarrow w^{\mathrm{c}}_{v'}$; $\mathrm{C_l}(w^{\mathrm{c}}_{v'}) \Leftarrow n^{\mathrm{c}}_{v'}$; $w^{\mathrm{c}}_{v'} \Leftarrow n^{\mathrm{c}}_{v'}$; $stack \Downarrow n^{\mathrm{c}}_v$
    end if
end loop

Algorithm 3.3 Convert binary tree $\mathcal{T}_v$ to spanning tree $\mathcal{T}_{v'}$

```
Initialize T_v' with node v as root
Initialize tag ⇐ 0 (tag ∈ {0, 1, 2})
stack ⇓ R(T_v)
w^c_v' ⇐ R(T_v')
if w^c_v' is leaf node then
  return;
else
  loop
    stack ⇑ n^c_v
    if n^c_v is null then
      if tag = 0 then
        stack ⇑; tag ⇐ 1
      else if tag = 1 then
        stack ⇑; tag ⇐ 2
      else if tag = 2 then
        return;
      end if
    else
      if tag = 0 then
        if C_l(n^c_v) is not null then
          stack ⇓ n^c_v; stack ⇓ C_l(n^c_v); n^c_v' ⇐ G_s(C_l(n^c_v)); P(n^c_v') ⇐ w^c_v';
          C_{N_c(w^c_v')}(w^c_v') ⇐ n^c_v'; w^c_v' ⇐ n^c_v'; tag ⇐ 0
        else
          stack ⇓ null; tag ⇐ 0
        end if
      else if tag = 1 then
        if C_r(n^c_v) is not null then
          stack ⇓ n^c_v; stack ⇓ C_r(n^c_v); n^c_v' ⇐ G_s(C_r(n^c_v)); P(n^c_v') ⇐ P(w^c_v');
          C_{N_c(P(w^c_v'))}(P(w^c_v')) ⇐ n^c_v'; w^c_v' ⇐ n^c_v'; tag ⇐ 0
        else
          stack ⇓ null; tag ⇐ 1
        end if
      else if tag = 2 then
        if n^c_v = C_l(P(n^c_v)) then
          tag ⇐ 1; w^c_v' ⇐ B_l(w^c_v')
        else
          tag ⇐ 2; w^c_v' ⇐ P(w^c_v')
        end if
        stack ⇑;
        if w^c_v' is null then
          return;
        end if
      end if
    end if
  end loop
end if
```

Algorithm 3.4 Convert binary tree $\mathcal{T}_v$ to a linear-element array a_e

Initialize a_e with null
$queue \Downarrow \mathrm{R}(\mathcal{T}_v)$
while $queue$ is not null **do**
  $queue \Uparrow n_v$
  $e_{a_e} \Leftarrow \mathrm{G_l}(n_v)$
  if $\mathrm{C_l}(n_v)$ is null **then**
    $\mathrm{c_l}(e_{a_e}) \Leftarrow$ '0'
  else
    $\mathrm{c_l}(e_{a_e}) \Leftarrow$ '1'; $queue \Downarrow \mathrm{C_l}(n_v)$
  end if
  if $\mathrm{C_r}(n_v)$ is null **then**
    $\mathrm{c_r}(e_{a_e}) \Leftarrow$ '0'
  else
    $\mathrm{c_r}(e_{a_e}) \Leftarrow$ '1'; $queue \Downarrow \mathrm{C_r}(n_v)$
  end if
  $a_e \Leftarrow (a_e \cup e_{a_e})$
end while

Algorithm 3.5 Convert linear-element array a_e to binary tree $\mathcal{T}_v$

Require: a_e is not null
Initialize $index = 0$; $m_1 = -1$; $m_2 = 0$
Initialize binary-node array a_n with null
$n_v \Leftarrow \mathrm{G_s}(a_e[index++])$
$\mathrm{R}(\mathcal{T}_v) \Leftarrow n_v$
$a_n[0] \Leftarrow \mathrm{R}(\mathcal{T}_v)$
while $m_1 \neq m_2$ **do**
  $cnt \Leftarrow 0$
  for $i = 1$ to $m2 - m1$ **do**
    $n_v^p \Leftarrow a_n[m_1 + i]$
    if $\mathrm{C_l}(n_v^p)$ is '1' **then**
      $n_v^c \Leftarrow \mathrm{G_s}(a_e[index++])$
      $\mathrm{C_l}(n_v^p) \Leftarrow n_v^c$; $\mathrm{P}(n_v^c) \Leftarrow n_v^p$; $a_n \Leftarrow a_n \cup n_v^c$; $cnt++$
    end if
    if $\mathrm{C_r}(n_v^p)$ is '1' **then**
      $n_v^c \Leftarrow \mathrm{G_s}(a_e[index++])$
      $\mathrm{C_r}(n_v^p) \Leftarrow n_v^c$; $\mathrm{P}(n_v^c) \Leftarrow n_v^p$; $a_n \Leftarrow a_n \cup n_v^c$; $cnt++$
    end if
    $m_1 = m_2$
    $m_2 = m_2 + cnt$
  end for
end while

Algorithm 3.6 Calculate the array of indexes to separate a_e into linear-trees

Require: a_e is not null
  Initialize *milestone_array* with null; $milestone\_cnt = 0$; $storage = 0$
  $milestone\_array[milestone\_cnt++] = 0$
  for $i = 0$ to $|a_e| - 1$ **do**
    if $c_l(a_e[i])$ **then**
      $storage++$;
    end if
    if $c_r(a_e[i])$ **then**
      $storage++$;
    end if
    if $storage = i$ **then**
      $milestone\_array[milestone\_cnt++] = i + 1$; $storage++$
    end if
  end for

3.2.3 Implementation of Reconstruction in Differential Update

In this part, we will talk about the algorithm which is used on the receiver side to reconstructs the neighbor's BFST in a differential update. The implementation of setting up a differential update can be more easily introduced after presenting the Algorithm 3.1, Algorithm 3.4, Algorithm 3.5, and Algorithm 3.6. By operating all four algorithms listed above, the linear-forest in a differential update can be separated into a set of linear-trees, and each linear-tree can be converted to a binary tree and further to a spanning tree. As we specified before, every edge in a differential spanning tree indicates a new parent in neighbor's BFST, hence the differential update can be easily handled by incorporating all spanning trees in the the local cached one. The incorporation process is similar with Algorithm 3.1, but there are two differences.

1. We do not check the hop count of path anymore, replace all edges with the new ones indicated by the differential spanning tree.

2. A spanning tree with root ID 255.255.255.255 is spatial case in the differential

update, all node in such spanning tree should be trimmed from the BFST of the neighbor.

3.3 Summary

In this chapter, we propose a new proactive source routing scheme with name PSR. PSR is a tree based routing scheme with hop as the metric, and every node in the network maintains a BFST tree to other nodes in the network which is rooted at the node itself. Hence, the source routing information can be used to provide the forwarder list for opportunistic data forwarding in MANETs, and we took a great deal of effort to reduce the routing overhead. A tree structure can provide both the topology information and the route costs together, and we use linearized the tree as the routing update to save routing overhead. We further propose to use less frequent full dump routing update and more frequent differential update to reduce the overhead. Then a greedy algorithm is put forward which is used to maintain the stable BFSTs between two iterations, so that the packet size of the differential update can be kept small. Last, based on our design, we present the pseudo-code of important algorithms in our implementations. In Chapter VI, we will see the overhead in PSR is only a small fraction or an order of magnitude smaller than the overhead in selected baselines.

Chapter 4

Topology Change Model and Large-scale Live Update

The mobility of nodes in MANETs is the most crucial feature differentiate MANETs from static wireless networks. In Chapter III, we have presented the routing module PSR. To be a proactive routing protocol, PSR requires nodes periodically exchange new topology information with their neighbors. Such a strategy is selected with the consideration that if the topology information is exchanged by the event driven but not driven by timer in a periodical way, too much overhead will be introduced, such as in the WRP [32] we introduced in Chapter II. However, the timer driven proactive routing protocol has a disadvantage that the farther away from source node to destination, the more inaccuracy we have in the routing knowledge. To deal with this issue, in this chapter we propose the large-scale quick update which can update the inaccurate routing information more quickly than just using routing update in PSR. Moreover, the large-scale quick update uses broadcast nature to finish its job, so no additional overhead or consumption will be introduced. In this chapter, we will first present a procedure to statistically evaluate the network topology changes, and then we will present the details of large-scale quick update.

4.1 Effect of Topology Change

Generally speaking, no matter which kind of protocols is used in MANET, the performance always decreases when the topology changes severely. In this section we will define term *topology change frequency* and show which parameters in the network can affect the topology change frequency in MANETs. The result in this section will be used to analyze the performance of CORMAN.

4.1.1 Assumptions and Definition

Here, we list the assumptions we made in our model:

i All nodes have the same maximum speed $\|\vec{\mathbf{v}}_M\|$, and every node i can move with velocity

$$\vec{\mathbf{v}}_i \in \{\vec{\mathbf{v}} | (0 \leq \|\vec{\mathbf{v}}\| \leq \|\vec{\mathbf{v}}_M\|) \,\&\&\, (0 \leq \angle(\vec{\mathbf{v}}) < 2\pi)\}. \tag{4.1}$$

In particular, both $\|\vec{\mathbf{v}}\|$ and $\angle(\vec{\mathbf{v}})$ follow uniform distribution, *i.e.*

$$\|\vec{\mathbf{v}}\| \sim U[0, \|\vec{\mathbf{v}}_M\|] \tag{4.2}$$

and

$$\angle(\vec{\mathbf{v}}) \sim U[0, 2\pi). \tag{4.3}$$

ii Nodes are homogeneously distributed in the network, and the planar density of nodes is ρ.

iii All nodes have the same transmission range r.

We define the topology change frequency as the percentage ratio of the changed links to all links in a unit time period. Formally, if we define $L^{+}_{\Delta\tau}$ and $L^{-}_{\Delta\tau}$ separately to denote the numbers of new established links and broken links during the time period $\Delta\tau$. Note that we assumed the nodes are homogeneously distributed, so the number of every node's neighborhood links is identical and the total number of links is a constant over time. Hence, we define L^{A} to denote the number of all undirected links in the network. Using these variables, the topology change frequency ΔE can be expressed as

$$\Delta E = \frac{L^{+}_{\Delta\tau} + L^{-}_{\Delta\tau}}{\Delta\tau} \times \frac{1}{L^{A}}. \tag{4.4}$$

Another ratiocination based on the second assumption is that when we consider the topology change frequency in a statistical way, we expect $L^{+}_{\Delta\tau} = L^{-}_{\Delta\tau}$. This can be easily proved. If $L^{+}_{\Delta\tau} \neq L^{-}_{\Delta\tau}$ then $\frac{L^{+}_{\Delta\tau}}{\Delta\tau} \neq \frac{L^{-}_{\Delta\tau}}{\Delta\tau}$, so after a period the distribution of nodes will not be homogeneous, which is contradictory with our assumption. Hence, we denote $\Delta L^{+/-}_{\Delta\tau}$ to present the value of $L^{+}_{\Delta\tau}$ and $L^{-}_{\Delta\tau}$, and the ΔE can be expressed as

$$\Delta E = \frac{2 \times \Delta L^{+/-}_{\Delta\tau}}{\Delta\tau} \times \frac{1}{L^{A}}, \tag{4.5}$$

and the topology break rate and topology establishment rate are both

$$\frac{\Delta L^{+/-}_{\Delta\tau}}{\Delta\tau} \times \frac{1}{L^{A}} = \frac{\Delta E}{2}. \tag{4.6}$$

As the result if we want to evaluate the ΔE, we just need to calculate the $L^{+}_{\Delta\tau}$ or $L^{-}_{\Delta\tau}$.

4.1.2 Topology Change Frequency Calculation

Considering that nodes can move simultaneously, we need to calculate the expectation relative velocity between them. Without loss of generality, we assume two nodes are A and B with vector velocity $\vec{\mathbf{v}}_a$ and $\vec{\mathbf{v}}_b$ respectively, and the angle between them is θ. According to previous assumptions, the direction of every node uniformly distributes from 0 to 2π, so the angle between them also follows the uniform distribution. Based on the analysis above, the expectation of the magnitude of relative velocity $E\left(\|\vec{\mathbf{v}}_r\|\right)$ can be calculated by

$$E\left(\|\vec{\mathbf{v}}_r\|\right) = \int_0^{2\pi}\int_0^{\|\vec{\mathbf{v}}_M\|}\int_0^{\|\vec{\mathbf{v}}_M\|}\sqrt{\|\vec{\mathbf{v}}_a\|^2+|\vec{\mathbf{v}}_b\|^2-2\|\vec{\mathbf{v}}_a\|\|\vec{\mathbf{v}}_b\|\cos(\theta)}\,\frac{\mathrm{d}\|\vec{\mathbf{v}}_a\|}{\|\vec{\mathbf{v}}_M\|}\,\frac{\mathrm{d}\|\vec{\mathbf{v}}_b\|}{\|\vec{\mathbf{v}}_M\|}\,\frac{\mathrm{d}\theta}{2\pi}. \tag{4.7}$$

Considering our first assumption every node i has velocity

$$\vec{\mathbf{v}}_i \in \{\vec{\mathbf{v}}|\left(0 \ \le \ \|\vec{\mathbf{v}}\| \ \le \ \|\vec{\mathbf{v}}_M\|\right)\&\&\left(0 \le \angle(\vec{\mathbf{v}}) < 2\pi\right)\}, \tag{4.8}$$

and so $\|\vec{\mathbf{v}}_i\|$ satisfies the relationship

$$\|\vec{\mathbf{v}}_i\| = \iota \times \|\vec{\mathbf{v}}_M\|\left(0 \le \iota \le 1\right), \tag{4.9}$$

and further more, we have relationship for θ

$$\theta = \vartheta \times 2\pi\left(0 \le \vartheta < 1\right). \tag{4.10}$$

After the substitution and simplification we have

$$E\left(\|\vec{\mathbf{v}}_r\|\right) = \|\vec{\mathbf{v}}_M\| \times \int_0^1 \int_0^1 \int_0^1 \sqrt{\alpha^2 + \beta^2 - 2\alpha\beta\cos(2\pi \times \vartheta)}\ \mathrm{d}\alpha\ \mathrm{d}\beta\ \mathrm{d}\vartheta. \tag{4.11}$$

The result of the triple integration is a constant. For the reason that we can not find the explicit expression of the integration result, we use Maple to find the constant $C_1 = 0.7249$, so $E\left(\|\vec{\mathbf{v}}_r\|\right) = 0.7249 \times \|\vec{\mathbf{v}}_M\|$. The physical meaning of the formula above is that the relative speed between any two nodes is expected as 0.7249 times the maximum speed. The orientation of the relative velocity can be any direction in the planet *i.e.* $0 \leq \angle(\vec{\mathbf{v}}_r) < 2\pi$. Therefore, we can transform a complex situation into a simple one, where one node can be taken as static, and the other node moves with the expected relative velocity $\vec{\mathbf{v}}_r$.

Continuing with the situation in Figure 4.1, where node A is taken as static node with transmission range r and the other node B, which moves in the coordinate system where A is taken as the original point, is s away from node A . As we analyzed before, the orientation of the relative velocity can be any one in the plant. We assume node B moves along the dash line toward A and the intersection point with the A's transmission boundary is point X. If we denote ϕ as the inner angle between segment AB and AX, the length of segment can be calculated by following expression which is a function of s and ϕ given by

$$L_s^\phi = \sqrt{s^2 + r^2 - 2sr\cos\phi}. \tag{4.12}$$

Also, because of the uniform distribution of θ, which is the angle between two velocities of two nodes, ϕ in Figure 4.1 also follows the uniform distribution. Therefore, the expectation of L_s from such a node B, which is s away from the center node A,

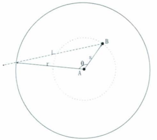


Figure 4.1: Sample figure to calculate the expectation of L_s

can be expressed as follows

$$E(L_s) = \int_0^{2\pi} \sqrt{s^2 + r^2 - 2sr\cos\phi}\,\frac{\mathrm{d}\phi}{2\pi}. \tag{4.13}$$

If we introduce another variable φ which satisfies $0 \leq \varphi < 1$ to substitute the ϕ above, and after the simplification we have

$$E(L_s) = \int_0^1 \sqrt{s^2 + r^2 - 2sr\cos(2\pi \times \varphi)}\,\mathrm{d}\varphi. \tag{4.14}$$

The physical meaning of the formula above is that statistically, if node is s away from another node, the node have to go L_s to get out of the transmission range of the latter one. Considering the expected magnitude of relative velocity $E(\|\vec{\mathbf{v}}_r\|)$, the expected time $E(t_s)$ used by a node, which is s away from the center node, to go out of the center node's transmission range is

$$E(t_s) = \frac{E(L_s)}{E(\|\vec{\mathbf{v}}_r\|)}. \tag{4.15}$$

Moreover, because the planar node density ρ is a constant over all area, the circular node density at the circle with radius s for every center node should be $2\pi s\rho$, and the number of nodes on a band with infinite small width Δs. Therefore, the partial topology break frequency contributed by these nodes can be presented as

$$\Delta E_A^s = \frac{2\pi s\rho \times \Delta s}{E\left(t_s\right)}. \tag{4.16}$$

Therefore, the final result of the topology break frequency contributed by node A in Figure 4.1 is

$$\Delta E_A^- = \int_0^r \frac{2\pi s\rho}{E\left(t_s\right)}\,\mathrm{d}s = \int_0^r \frac{2\pi s\rho}{\frac{\int_0^1 \sqrt{s^2+r^2-2sr\cos(2\pi\times\varphi)}\,\mathrm{d}\varphi}{E(\|\vec{\mathbf{v}}_r\|)}}\,\mathrm{d}s. \tag{4.17}$$

As well, we use a trick to simplify the integral above. Assume we have another variable ς which satisfies the constraint that $0 \leq \varsigma \leq 1$, hence we have $s = \varsigma \times r$. Substitute in the integral above and simplify the result, and finally we have

$$\Delta E_A^- = 2\pi r\rho \times E\left(\|\vec{\mathbf{v}}_r\|\right) \times \int_0^1 \frac{\varsigma}{\int_0^1 \sqrt{\varsigma^2 - 2\varsigma\cos\left(2\pi\times\varphi\right)+1}\,\mathrm{d}\varphi}\,\mathrm{d}\varsigma. \tag{4.18}$$

Unfortunately, the denominator of the integral in the formula above is an *Elliptic Integral*, which is similar to

$$\int_0^1 \sqrt{\varsigma^2 - 2\varsigma\cos\left(2\pi\times\varphi\right)+1}\,\mathrm{d}\varphi = \frac{2\left(1+\varsigma\right) EllipticE\left(\frac{2\sqrt{\varsigma}}{\varsigma+1}\right)}{\pi}, \tag{4.19}$$

where

$$EllipticE\left(\frac{2\sqrt{\varsigma}}{\varsigma+1}\right) \tag{4.20}$$

is the *Second Kind Complete Elliptic Integral*, which can only be expressed as a power series as follows

$$EllipticE\left(\frac{2\sqrt{\varsigma}}{\varsigma+1}\right) = \frac{\pi}{2}\sum_{n=0}^{\infty}\left[\frac{(2n)!}{2^{2n}n!^2}\right]^2 \frac{\left(\frac{2\sqrt{\varsigma}}{\varsigma+1}\right)^{2n}}{1-2n}. \tag{4.21}$$

We use Maple to calculate the integral about ς, and its value is nearly 0.4439. Therefore the formula of ΔE_A^- can be written as

$$\Delta E_A^- \simeq 0.4439 \times 2\pi r\rho \times E\left(\|\vec{\mathbf{v}}_r\|\right). \tag{4.22}$$

The result has the unit *number of nodes per second.* If we have a network composed by N nodes, and these nodes can move freely in the S planar area, we have

$$\rho = \frac{N}{S}. \tag{4.23}$$

Therefore, the number of neighbors for every node in the network can be expressed as

$$n = \pi r^2 \frac{N}{S} - 1. \tag{4.24}$$

The number of undirected links L^A in the network is

$$L^A = \frac{N \times n}{2}, \tag{4.25}$$

and the link break frequency all over the network is

$$\frac{L^-_{\Delta\tau}}{\Delta\tau} = \frac{N \times \Delta E_A^- \times \Delta\tau}{\Delta\tau} = N \times \Delta E_A^-. \tag{4.26}$$

Finally, by the definition which is in the beginning of this section, the topology change frequency ΔE can be calculated by

$$\Delta E = \frac{2 \times \frac{L_{\Delta\tau}^{-}}{\Delta\tau}}{L^{A}} = 2 \times N \times \Delta E_{A}^{-} \times \frac{2}{N \times n} = \frac{2.5743 \times \pi r \frac{N}{S} \times \|\vec{\mathbf{v}}_{M}\|}{\pi r^{2} \frac{N}{S} - 1}. \tag{4.27}$$

When the node density is quite high, the formula above can be simplified as

$$\Delta E \approx 2.5743 \frac{\|\vec{\mathbf{v}}_{M}\|}{r}. \tag{4.28}$$

If convert unit to *percentage per second* (%/s), the result above can be presented by

$$\Delta E \approx 257.43 \frac{\|\vec{\mathbf{v}}_{M}\|}{r} \ (\%/s), \tag{4.29}$$

which means the topology changes $257.43 \frac{\|\vec{\mathbf{v}}_{M}\|}{r}$ percent every second.

4.2 Large-scale Live Update

When a batch of packets are forwarded along the route towards the destination node, if an intermediate node is aware of a new route to the destination, it is able to use this new route to forward the packets that it has already received. There are a few implications of this. First, this new route will also be used to forward the subsequent packets of the same batch. Second, when packets are forwarded along the new route, such an updated forwarder list replaces the old list in the packets. As a result, the upstream nodes can be notified of the new route and this information can propagate back to the source node quickly. Details of data forwarding and list update are described in this section with the help of Figure 4.2. In the figure, the source node v_1 has a flow of data packets for destination node v_{10}. According to its own routing module, v_1 decides that the best route to v_{10} is $v_1v_2v_3v_4v_5v_6v_7v_8v_9v_{10}$; hence the forwarder list.

At a given point of time during the data transfer of a batch, there is a node on the forwarder list that has the highest priority and has received any packet of the batch. We call such a node the *frontier* of the batch. At the beginning, the frontier is the source node. When the destination has received at least one packet of a batch, it has become the frontier of the batch. Recall that a fragment (Section 2.1.1.2) is a subset of packets in the current batch which are sent together from a given forwarder. Here, the frontier has cached its first fragment of packets. Suppose at this point, the frontier in Figure 4.2 is v_3. When it is about to forward this fragment, if its routing module indicates that there is a new route to the destination, *e.g.*, $v_3v_4v_5v_6'v_7'v_9v_{10}$, it replaces the segment of the original forwarder list from itself to the destination (*i.e.*, $v_3v_4v_5v_6v_7v_8v_9v_{10}$) with this new route. That is, the forwarder list carried by these data packets are now $v_1v_2v_3v_4v_5v_6'v_7'v_9v_{10}$. When the packets of the fragment are forwarded, they will be following the new route. In addition, upstream nodes can overhear these packets, and thus their new forwarder list. These nodes can update their own routing information and will incorporate such information when forwarding their fragments. This backtrack continues until the source is aware of the latest route information.

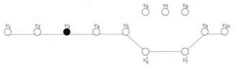


Figure 4.2: Route update

We would like to bring up the following notes to the readers' attention.

1. When the network diameter is large and nodes are moving fast, the routing information can be obsolete by the time it has propagated to a remote node. That is, a node's knowledge about the network topology becomes less accurate when the destination node is located farther away. Thus, the forwarder list composed by the source node needs to be adjusted as packets are forwarded towards the destination, where intermediate nodes closer to the destination could have better routing information. This is achieved effectively by allowing the frontier node to modify the forwarder list carried by the fragment of packets. As a result, CORMAN has a fairly good tolerance of route inaccuracy for any source node to start with.

2. When a frontier node updates a forwarder list, only the segment of the list between the frontier and destination is replaced while the rest of the list (*i.e.*, nodes that the fragment have gone through) are intact. The reason for this design decision is that these upstream nodes should not be disturbed by the new route so that the scheduling coordination among them is consistent.

3. We only allow a frontier node to update a packet's forwarder list according to its routing module. A node that is no longer a frontier should only incorporate the forwarder list that it overhears from downstream nodes. The purpose is to avoid unnecessary updates of route information as the time needed to transfer a batch of data packets is very short. During this time, usually little has changed about the network topology and nodes may not even have exchanged the routing information for the next periodic interval.

4. Consider a particular intermediate node on the forwarder list. As the frontier moves from the source to the destination, the forwarder list may be refreshed multiple times by different frontiers. Thus, this node may experience one update

about its route to the destination every time a frontier decides to modify the list.

All of the above is achieved rapidly and with no extra communication overhead compared to ExOR.

4.3 Summary

In this chapter, we first calculate the topology changes in percentage. In particular, the percentage is directly related the average velocity of mobile nodes and inversely related with wireless transmission range. The large-scale live update is proposed to reduce the effect of topology changes in mobile networks and we put forward the scheme based on two considerations. On one hand, because the proactive source routing scheme we proposed in Chapter III has a feature that all routing information can only be shared with its one hop neighbor periodically, the topology information maintained between a long route may not be accurate as it is always changing in mobile networks. One the other hand, in opportunistic data forwarding, the nodes that are closer to destination will access wireless media more aggressively to forward its received packet, and the nodes that are closer to destination will have fresher topology information from it to destination, and moreover, all packets in opportunistic data forwarding contain the forwarder list which is composed by nodes on the entire route. Hence, in the large-scale live update we designed, the downstream frontiers will update the forwarder list of pioneer packets, and so the upstream node can update the downstream topology more quickly by overhearing these pioneer packets without any additional overhead.

Chapter 5

Small-scale Retransmission

As we mentioned before, implementation of opportunistic date transfer over the least hops path can explore the broadcast nature in wireless transmission, However the penalty for using such path to transmit packets is that the connectivity between every pair of forwarder may be vulnerable. We propose a solution to solve this problem, and the basic idea is that the nodes which is not contained in the forwarder list [9] can run a distributed *small-scale retransmitter selection* algorithm to verify whether it should participate in the packet forwarding. Such a small-scale retransmitter should have the *best* position between two *listed forwarders* (the forwarders which are contained in the forwarder list) compared with other nodes. In general, our small-scale retransmitter selection algorithm has three important features. First, it is a distributed algorithm. In particular, if there are many nodes between two listed forwarders, after every node running the small-scale retransmitter selection algorithm, the best node will know it is the best, and participate in the data forwarding without sending any announcement. Meanwhile, other nodes will automatically refuse to be the small-scale retransmitter without any notification either. Second, the small-scale retransmitter

The research findings from small-scale retransmission has been submitted to IEEE ICC'12 [42] selection algorithm is operated when the packets in a batch are received by the nodes which are not the forwarders in the forwarder list, so the small-scale retransmitter can be selected on time without the problem of inaccuracy. Third, all information required by the node to run such as algorithm has already been collected by routing packets exchanging, so no additional information needs to be injected into network. In this chapter, we will first extensively analyze the reason why the small-scale retransmitter is required in CORMAN and then specify the details in our solution.

5.1 Considerations of Small-scale Retransmission

Previously, we proposed that the forwarder list should be updated in a piggyback way when data packet is being forwarded toward the destination. However, it is not helpful to enforce the link connectivity, because updating the forwarder list only helpful when there is another node replaces the moving away one, but if no node replaces the position, the forwarder list will still be broken, which is shown in Figure 5.1. Node Y moves from the position Y' to the current position with the direction shown by the dashed arrow. Even though the wireless link is quite vulnerable, a transient link from Y to X may exist and over such a link, node X sends downstream nodes the information that the precursor to go to node Y is X. However, the vulnerable link is unreliable or broken when the source node wants to use forwarder list "$Dest, Y, X, C, B, A, Src$", even by opportunistic data forwarding. To deal with this problem, we propose that

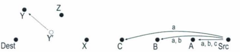


Figure 5.1: Situation which can not be solved by forwarder list update

node Z, which is a neighbor for both nodes X and Y, can be the small-scale retransmitter to offer a help when it receives packets from upstream nodes. That is because if node Z receives a packet with forwarder list "$Dest, Y, X, C, B, A, Src$" Z can find Y and X are assigned forwarders and itself is a neighbor for both of them. Hence node Z could participate to forward data packet. In particularly, node Z also waits for a period of time before its forwarding, such a time period should be longer than the waiting time of node Y but shorter than the waiting time of node X. Or if Z can overhear a fragment of packets being transmitted from node Y, it can predict the exactly time when Y will finish its transmission and begin forwarding rest of the useful packet directly. However, before the discussion of the details of the small-scale retransmitter selection algorithm, it is necessary to introduce the changes to the data structure kept on each node.

5.2 Design of Small-scale Retransmission

In general, a *Neighbor Table* in CORMAN maintains the records of two hop topology of current node and the link quality from its neighbors to its neighbors' neighbors. In CORMAN, we use *Received Signal Strength Indicator* (RSSI) to evaluate the quality of links, because according to the work done by Charles Reis *et al.* [43] and Mei-Hsuan Lu *et al.* [44], RSSI can be reported by almost all wireless cards and a greater RSSI value usually indicates a better wireless channel quality.

In particular, *RSSI Table* is a sub-table maintained in every *Neighbor Table* entry which records the RSSI values from that neighbor to that neighbor's neighbors, and the entries in the *RSSI Table* has the content shown in Figure 5.2. The *Neighbor's Neighbor ID* keeps the address of the nodes which are in fact two hop neighbors via current neighbor in *Neighbor Table*; the *RSSI* records the RSSI value from current neighbor to such two hop neighbors; the *Expire Time* holds the time when the entry should be removed; and the *next* helps us find next entry in the same *RSSI Table*.

Neighbor's Neighbor ID	RSSI	Expire Time	*next*→

Figure 5.2: Entry of RSSI Table in Neighbor Table entry

The operations on the *Neighbor Table* are presented as follows. When a routing update packet needs to be transmitted out from one node periodically, it not only contains the routing information but also collects the RSSI values from all its local *Neighbor Table* entries. All tuples $(Neighbor\ ID, RSSI)$ are added into the routing packet as the *Neighbor RSSI List*. After the node sending out such a packet, all its neighbors update the *RSSI Tables* kept by the corresponding *Neighbor Table* entries.

To make the following discussion easier, we give a brief topology in Figure 5.3. In

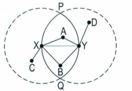


Figure 5.3: Topology example

the topology shown by Figure 5.3, we assume the wireless link from node X to node Y is a transient link, and such a vulnerable link may successfully deliver a routing packet from X to Y or from Y to X. Hence, the link between X and Y may used as a part of the forwarder list by a node far away. In this topology we can see the neighbors of node X are A, B, C and Y, and node Y's neighbors are A, B, D and X. Therefore, nodes X and Y have two same neighbors A and B. We use $R(M, N)$ to denote the RSSI value detected by node N from node M's transmission, and to make a intuitional discussion, the *Neighbor Tables* and their *RSSI Tables* on both node A and B are briefly presented in Figure 5.4.

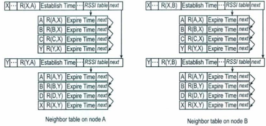


Figure 5.4: Neighbor table on node A and node B

Without loss of generality, in the Figure 5.3 we assume node A and node B overhears one or more packets in a same batch, and node X and node Y are contained in the forwarder list in these packets. In traditional opportunistic data forwarding, only the nodes in the forwarder list can participate in the packet forwarding process, and the probability that the packet transmitted between X and Y may be quite low. That is because the least hop path may contain unreliable and transient links in the forwarder list, and such situation becomes even worse in the mobile case. Such a problem can be solved by choosing one node from A and B as the small-scale retransmitter which is selected by small-scale retransmitter selection algorithm as we present below.

5.3 Algorithm and Scoring Function

In particular, node A and B will decode the forwarder list from the overheard packet, if they are not contained in the forwarder list (which is true in our assumption, otherwise the node will participate to forward packets certainly), both of them will check whether it is a neighbor node of two adjacently listed forwarder pair in the forwarder list. This work can be done easily by a searching algorithm with complexity $O(n \times m)$, where n is the number of neighbors of A or B, and m is the length of the forwarder list.

In our example shown in Figure 5.3, both A and B are the neighbors of X and Y, and we should evaluate further like follows. Take node A as an example, it will check whether the $R(A, X)$ is greater than $R(Y, X)$ and whether the $R(A, Y)$ is greater than $R(X, Y)$. If we assume links are symmetric, the $R(X, Y)$ should equal $R(Y, X)$, and such four RSSI values are maintained in the *Neighbor Table.* If both of previous two comparisons are true, node A knows it is a *valid* small-scale retransmitter, but can not guarantee it is the *best* one. To check whether it is the *best* small-scale retransmitter, it will lookup whether other valid small-scale retransmitters exist in the network. This can be done by running a simple searching algorithm on the listed forwarders' *RSSI Tables*. In our example, we should search the same *Neighbor's Neighbor ID* from X's and Y's *RSSI Tables* in A's *Neighbor Table* and the complexity is $O(n_X \times n_Y)$, where n_X and n_Y are the number of neighbors of nodes X and Y. In our example, another valid small-scale retransmitter would be B if $R(B, X)$ is greater than $R(Y, X)$ and the $R(B, Y)$ is greater than $R(X, Y)$ are both true.

If B is also a valid small-scale retransmitter judged by node A, node A has to make a decision which one (A itself or B) is better. We propose a scoring function given below to evaluate the score of every valid small-scale retransmitter i between M and N.

$$\begin{aligned} W(i) = \left(\sqrt[4]{\frac{1}{R(i,N)}} + \sqrt[4]{\frac{1}{R(i,M)}}\right) \times \\ \left(\frac{1}{4} \times \ln \frac{\max\left(\frac{1}{R(i,N)}, \frac{1}{R(i,M)}\right)}{\min\left(\frac{1}{R(i,N)}, \frac{1}{R(i,M)}\right)} + \ln\left(\sqrt[4]{\frac{1}{R(i,N)}} + \sqrt[4]{\frac{1}{R(i,M)}}\right)\right). \end{aligned} \tag{5.1}$$

Hence, A has to compare the score of itself, given by

$$W(A) = \left(\sqrt[4]{\frac{1}{R(A,Y)}} + \sqrt[4]{\frac{1}{R(A,X)}}\right) \times \left(\frac{1}{4} \times \ln \frac{\max\left(\frac{1}{R(A,Y)}, \frac{1}{R(A,X)}\right)}{\min\left(\frac{1}{R(A,Y)}, \frac{1}{R(A,X)}\right)} + \ln\left(\sqrt[4]{\frac{1}{R(A,Y)}} + \sqrt[4]{\frac{1}{R(A,X)}}\right)\right) \tag{5.2}$$

to the score of node B by given by

$$W(B) = \left(\sqrt[4]{\frac{1}{R(B,Y)}} + \sqrt[4]{\frac{1}{R(B,X)}}\right) \times \left(\frac{1}{4} \times \ln \frac{\max\left(\frac{1}{R(B,Y)}, \frac{1}{R(B,X)}\right)}{\min\left(\frac{1}{R(B,Y)}, \frac{1}{R(B,X)}\right)} + \ln\left(\sqrt[4]{\frac{1}{R(B,Y)}} + \sqrt[4]{\frac{1}{R(B,X)}}\right)\right), \tag{5.3}$$

and the smaller one should be chosen as the result of small-scale retransmitter selection algorithm. The same comparison will also be operated on node B because node B knows node A is a valid small-scale retransmitter too.

The algorithm does not need additional packets to coordinate the final decision on different nodes because the *RSSI Tables* kept for node X and Y on node A and the *RSSI Tables* kept for node X and Y on node B must be the same. That is because if A and B are both neighbors for both X and Y, the RSSI information broadcasted from X and Y will be received simultaneously by node A and B and refresh their corresponding *RSSI Tables* together. Therefore, every valid small-scale retransmitters will finally *agree* on a particular one as the *best* small-scale retransmitters out of themselves without gossip.

An important explanation must be given here which is why we propose Formula 5.1 to evaluate the valid small-scale retransmitters. Even though the RSSI calculation method is not specified in the 802.11 standard, 802.11 device vendors should calculate the RSSI according to the received signal strength, and the higher the received signal strength, the greater the RSSI value that should be returned from hardware[45]. Ideally, the path loss model of wireless communication follows the rule that the received signal power is inversely proportional to d^n, where d is the distance between the transmitter and the receiver and n is usually an integer within 2, 3, and 4. If we take other parameters as constants, the ideal relationship between distance from M to N and RSSI can be presented as

$$R(M,N) = C \times d_{M,N}^{-n} \tag{5.4}$$

where C is a constant.

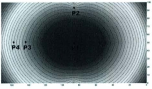
When we want to get the reciprocal of RSSI we have

$$\frac{1}{R(M,N)} = \frac{d_{M,N}^{n}}{C} \tag{5.5}$$

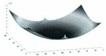
The rationale for comparing scores calculated by Formula 5.1 explained as follows. Considering Formula 5.5, the Formula 5.1 can be simplified to Formula 5.6,

$$\begin{aligned} W(i) &= \left(\frac{d_{i,M}^{\frac{n}{4}}}{\sqrt[4]{C}} + \frac{d_{i,N}^{\frac{n}{4}}}{\sqrt[4]{C}}\right) \times \left(\ln \frac{\max\left(\frac{d_{i,M}^{\frac{n}{4}}}{\sqrt[4]{C}}, \frac{d_{i,N}^{\frac{n}{4}}}{\sqrt[4]{C}}\right)}{\min\left(\frac{d_{i,M}^{\frac{n}{4}}}{\sqrt[4]{C}}, \frac{d_{i,N}^{\frac{n}{4}}}{\sqrt[4]{C}}\right)} + \ln\left(\frac{d_{i,M}^{\frac{n}{4}}}{\sqrt[4]{C}} + \frac{d_{i,N}^{\frac{n}{4}}}{\sqrt[4]{C}}\right)\right) \\ &= \frac{\left(d_{i,M}^{\frac{n}{4}} + d_{i,N}^{\frac{n}{4}}\right)}{4 \times \sqrt[4]{C}} \times \left(n \times \ln \frac{\max(d_{i,M}, d_{i,N})}{\min(d_{i,M}, d_{i,N})} + 4 \times \ln\left(d_{i,M}^{\frac{n}{4}} + d_{i,N}^{\frac{n}{4}}\right) - \ln C\right) \end{aligned} \tag{5.6}$$

We can see that a node which leads the minimum value of Formula 5.6 should be the node which has nearly the same distances to both node M and node N, and should has the least summation of two such distances. The parameters in Formula 5.1 are selected by a great deal of trials in Matlab. And finally, the figures we created in Matlab when n equals 2, 3 and 4 are shown in Figures 5.5, 5.6, and 5.7.

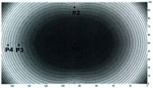


(a) Contour line from top

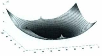


(b) Profile

Figure 5.5: Scoring function with $n = 2$ as the path loss parameter

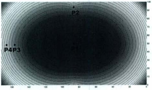


(a) Contour line from top

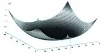


(b) Profile

Figure 5.6: Scoring function with $n = 3$ as the path loss parameter



(a) Contour line from top



(b) Profile

Figure 5.7: Scoring function with $n = 4$ as the path loss parameter

In Figures 5.5, 5.6, and 5.7, the rectangle with the Width/Height ratio $\sqrt{3}$ is the $\pi/2$ rotated circumscribed quadrilateral of the olivary shape in Figure 5.3 which is surrounded by $\widehat{PXQ}$ and $\widehat{PYQ}$. The node X and Y are located on the position $(50 \times \sqrt{3}, 0)$ and $(50 \times \sqrt{3}, 100)$, and the node P and Q are located on the position $(0, 50)$ and $(100 \times \sqrt{3}, 50)$. The contour line is the curve that all nodes on which have the same score given by Formula 5.6. We can see that, for all possible n from 2 to 4, the contour lines follow the same rule that when a node is nearer to one of two connected neighbor, the node's score is greater, so it has less opportunity to be selected. That is reasonable because the small-scale retransmitter which is too near with one of the two nodes that connected by a vulnerable link has less contribution than that in the middle of such two nodes. Furthermore, the center point on every figure always has the minimum value, which indicates to us where the best small-scale retransmitter is supposed to be.

For example P_2 in all the above figures probably has less contribution than P_1 to be as a small-scale retransmitter. One more thing, P_2 should have a better position than P_4 because the P_4 is near the position of P or Q in Figure 5.3 and has longer distance to both node X and Y than P_2, and from our scoring function it does have a smaller score than P_4. Hence, the contribution made by sender diversity and receiver diversity from P_2 is more than the contribution given by P_4. Furthermore, considering P_2 is better than P_4 and worse than P_1, and P_4, P_1 are on the same center line, we predict there must be a point on the center line which has the same score with P_2. In fact, we do find such a point P_3 which is on the same contour line of P_2.

Based on the analysis before, we can see that the scoring function we proposed matches our predictions very well, so we proved the function is suitable to be used to evaluate valid small-scale retransmitters.

5.4 Summary

In this chapter we proposed the small-scale retransmission. From the evolution point of view, the small-scale retransmission extend the opportunistic data forwarding one step further than ExOR. That is because ExOR utilize the benefit of the broadcast nature by adding all nodes on the route, the forwarder list, into a packet, and any one that receives the packet could forward it. However, the nodes that are not contained in the forwarder list can not give a hand even if they are on the right direction from source to destination. Furthermore, the transient high quality links may quite easily break in mobile networks, and the small-scale retransmission we proposed in this chapter can effectively bridge the broken links and maintain a robust topology for opportunistic data forwarding. To implement the small-scale retransmission, we require every node to maintain the RSSI information on links within two hops, and only use the best retransmitter in a proper region between listed forwarders to help us with opportunistic data forwarding. Moreover, to select the best node and coordinate all feasible retransmitter-candidates, a scoring function is proposed in this chapter as well. As a result, nodes with two hop RSSI information can independently make an identical decision on selecting the best small-scale retransmitter, with no additional coordination overhead.

Chapter 6

Performance Evaluation

In this section, we will study the performance of the proposed solution that enable the opportunistic data forwarding in MANETs. In particular, we not only study the overall performance where all the solutions work together, but also study the particular effectiveness of PSR (Chapter III) and the small-scale retransmission (Chapter V). Hence, this chapter is composed by three sections: the performance study of PSR, the effectiveness study of small-scale retransmission, and the overall performance of CORMAN.

6.1 Performance Study of PSR

We study the performance of PSR using computer simulation with Network Simulator 2 [46] (version 2.34). We compare PSR against OLSR [4], DSDV [3], and DSR [5], three fundamentally different routing protocols in MANETs, with varying network densities and node mobility rates. We measure the data transportation capacity of these protocols supporting TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) with different data flow deployment characteristics. Our tests show that the overhead of PSR is indeed only a fraction of that of the baseline protocols. Nevertheless, as it provides global routing information at such a small cost, PSR offers similar or even better data delivery performance. In this section, we first describe how the experiment scenarios are configured and what measurements are collected. Then we present and interpret the data collected from networks with heavy TCP flows and from those with light UDP streams.

6.1.1 Experiment Settings

We use the Two-Ray Ground Reflection channel propagation model in our simulation. Without loss of generality, we select a 1Mbps nominal data rate at the 802.11 links to study the relative performance among the selected protocols. With the default Physical Layer parameters of the simulator, the transmission range is approximately 250m and the carrier sensing range is about 550m.

We compare the performance of PSR with that of OLSR, DSDV, and DSR. The reasons why we select these baseline protocols that are different in nature are as follows. On one hand, OLSR and DSDV are both proactive routing protocols and PSR is also in this category. On the other hand, OLSR makes complete topological structure available at each node, whereas in DSDV, nodes only have distance estimates to other nodes via a neighbor. PSR sits in the middle ground, where each node maintains a spanning tree of the network. Furthermore, DSR is a well accepted reactive source routing scheme and, as with PSR, it support source routing, which does not require other nodes to maintain forwarding lookup tables. For a more leveled comparison to the peer proactive protocols, we make PSR transport regular IP packets rather source routed ones although it can and should be used to as a source routing scheme. All three baseline protocols are configured and tested out-of-the-box of ns-2.

In modeling node mobility of the simulated MANETs, we use the Random Waypoint Model to generate node trajectories. In this model, each node moves towards a series

of target positions. The rate of velocity for each move is uniformly selected from $[0, v_{\max}]$. Once it has reached a target position, it may pause for a specific amount of time before moving towards the next position. All networks have 50 nodes in our tests. We have two series of scenarios based on the mobility model. The first series of scenarios have a fixed $v_{\max}$ but different network densities by varying the network dimensions. The second series have the same network density but varying $v_{\max}$.
We study the data transportation capabilities of these routing schemes and their overhead in doing so by loading the networks with TCP data flows and UDP voice streams.

- To test how TCP is supported, in each scenario, we randomly select 40 nodes out of the 50 and pair them up. For each pair, we set up a permanent one-way FTP (File Transfer Protocol) data transfer. We repeat the selection of the 40 nodes five times and study their collective behavior. This essentially mimics heavily loaded mobile networks. For all four protocols, we measure their TCP throughput, end-to-end delay, and routing overhead in byte per node per second in each scenario.

- To study their performance in supporting UDP, we use two-way Constant Bit Rate (CBR) streams for compressed voice communications. Specifically, we select 3 pairs of nodes and feed each node with a CBR flow of 160 byte/pkt and 10 pkt/s, which simulates mobile networks with a light voice communication load. We measure the Packet Delivery Ratio (PDR), end-to-end delay, and delay jitter in each scenario.

Results about TCP (Sections 6.1.2 and 6.1.3) and UDP (Sections 6.1.4 and 6.1.5) with regard to varying node densities and velocity rates are in the subsequent subsections.

6.1.2 TCP with Node Density

We first study the performance of PSR, OLSR, DSDV, and DSR in supporting 20 TCP flows in networks with different node densities. Specifically, with the default 250m transmission range in ns-2, we deploy our 50-node network in a square space of varying side lengths that yield node densities of approximately $5, 6, 7, \ldots, 12$. These nodes move following the random waypoint model with $v_{\max} = 30$ m/s.

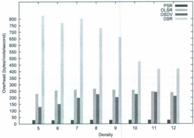


Figure 6.1: Routing overhead vs. density

We plot in Figure 6.1 the per-node per-second routing overhead, *i.e.*, the amount of routing information transmitted by the routing agents measured in byte/node/second, of the four protocols when they transport a large number of TCP flows. This figure shows that the overhead of PSR (20 to 30) is just a fraction of that of OLSR and DSDV (140 to 260) and more than an order of magnitude smaller than DSR (420 to 830). The routing overhead of PSR, OLSR, and DSDV goes up gradually as the node density increases. This is a typical behavior of proactive routing protocols in MANETs. Such protocols usually use a fixed time interval to schedule route exchanges. While the number of routing messages transmitted in the network is always constant for a given network, the size of such message is determined by the node density. That is, a node periodically transmits a message to summarize changes as nodes have come into or gone out of its range. As a result, when the node density is higher, a longer update message is transmitted even if the rate of node motion velocity is the same. Note that when the node density is really high, say around 10 and 12, the overhead of OLSR flattens out or even slightly decreases. This is a feature of OLSR when its Multi-Point Relay mechanism becomes more effective in removing duplicate broadcasts, which is the most important improvement of OLSR over conventional link-state routing protocols. PSR uses a highly concise design of messaging, allowing it to have a much smaller overhead than the baseline protocols. In contrast, DSR as a reactive routing protocol incurs a significantly higher overhead when transporting a large number of TCP flows because every source node needs to conduct its own route search. This is not surprising as reactive routing protocol were not meant to be used in such scenarios. Later in our experiments (Sections 6.1.4 and 6.1.5), we test all four protocols supporting just a few UDP streams for a different perspective. Here, the routing overhead of DSR decreases with the node density going up and network diameter going down. This is because the number of hops to a destination is smaller in a denser network, so the shorter, more robust routes break less frequently and do not need as many route searches. Furthermore, compared to IP forwarding, the fact that DSR is source routing and that intermediate nodes cannot modify the routes embedded in data packets works against its performance in a mobile network, both in terms of the increase of search operations and the loss of data transportation capacity. The reason is that, because a source node can be quite a few hops away from the destination, its knowledge about the path as embedded in the packets can become obslete quickly in a highly mobile network. As a packet progresses en route, if an intermediate node cannot reach the next hop as indicated in the embedded path, it will be dropped. This is very different from IP forwarding, where intermediate nodes can have more updated routing information than the source and can utilize that information in forwarding decisions.

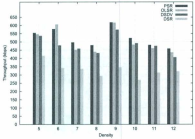


Figure 6.2: TCP throughput vs. density

Figure 6.2 plots the TCP throughput of the four protocols for the same node density levels as before. The total throughput of the 20 TCP flows of PSR, OLSR, and DSDV is noticeably higher than that of DSR. In addition, while the TCP throughput of DSR decreases with node density, that for the other three are somewhat unaffected, hovering at around 500kbps. Apparently, the large routing overhead of DSR, especially in dense networks, consumes a fair amount of channel bandwidth, leaving less room for data transportation. In most cases, PSR has the highest throughput because it needs to give up the least network resources for routing.

Next, we focus on the end-to-end delay of TCP flows to investigate how well these protocols support time-sensitive applications. Figure 6.3 presents the delay measured for different node densities. As the density increases from 5 to 12 neighbors, the delay of DSR goes up from 0.58s to about 1.5s, which is significantly higher than the typical value of 0.15s to 0.35s for the other three protocols. Such a difference is caused by the

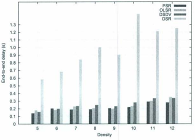


Figure 6.3: End-to-end delay in TCP vs. density

initial route search when a TCP flow starts and by the subsequent searches triggered by route errors. As the network becomes denser, all protocols show an increasing trend in end-to-end delay. This may seem counter-intuitive as, in denser networks, the average hop distance between source-destination pairs is smaller, which should lead to shorter round-trip time. However, this benefit is completely offset by more intense channel contention. Recall that the node density is inversely proportional to the square of network diameter. As such, in the interplay between route length and channel contention, the latter dominates the overall effect.

6.1.3 TCP with Velocity

We also study the performance of PSR and compare it to OLSR, DSDV, and DSR with different rates of node velocity. In particular, we conduct another series of tests in networks of 50 nodes deployed in a 1100×1100 (m^2) square area with $v_{\max}$ set to $0, 4, 8, 12, \ldots, 32$ (m/s). The network thus has an effective node density of around 7 neighbors per node, *i.e.*, a medium density among those configured in the previous subsection. As with before, 20 TCP one-way flows are deployed between 40 nodes and we measure the routing overhead, TCP throughput, and end-to-end delay (Figures 6.4, 6.5, and 6.6).

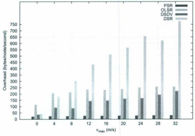


Figure 6.4: Routing overhead vs. velocity

The routing overhead of all four protocols with varying rates of node velocity is plotted as in Figure 6.4. Note that the velocity to the right of the x-axis corresponds to the middle bars in Figure 6.1. We observe in the plot here, as $v_{\max}$ decreases, the overhead of all protocols comes down. The reason for DSR is that, as the network structure becomes more stable, fewer route repair attempts are necessary. For the case of the proactive protocols, it is the reduction in the size of routing messages (*i.e.*, fewer neighbors have changed positions) that cuts down the overhead. Still relative among these four protocols, when the network is not stationary ($v_{\max} \neq 0$), the overhead of PSR (20 to 30 byte/node/second) is a fraction of that of OLSR and DSDV (90 to 300), and more than an order of magnitude lower than DSR (180 to 770).

The TCP throughput and end-to-end delay are plotted in Figures 6.5 and 6.6 respectively. From these figures, we observe that the performance of PSR, OLSR, and DSDV are similar with PSR leading the pack in most cases. In addition, neither throughput

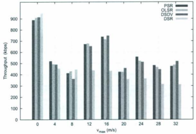


Figure 6.5: TCP throughput vs. velocity

or delay is affected by the different rates of velocity. The only exception is that when $v_{\max} = 0$, all protocols yield a high throughput of 900kbps. With a greater portion of the channel bandwidth devoured by routing messages in highly mobile networks, DSR suffers a noticeable performance penalty in TCP throughput and end-to-end delay.

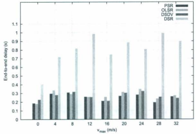


Figure 6.6: End-to-end delay in TCP vs. velocity

6.1.4 UDP with Density

We also tested the four protocols for their performance in transporting a small number of UDP streams. This is a typical assumption for ideal scenarios of reactive routing protocols. Here, we deploy three two-way UDP streams in order to simulate compressed voice communications. To find out about how node density affects these protocols, we use the same network and mobility configurations as in Section 6.1.2. We measure and plot the PDR (Figure 6.7), delay (Figure 6.8), and delay jitter (Figure 6.9) against varying node densities.

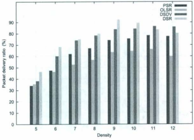


Figure 6.7: PDR in UDP vs. density

In Figure 6.7, the PDRs of all four protocols are in the same ball park across different node densities, with DSR slightly in the lead and OLSR trailing behind. This verifies that the traffic configuration is favorable for DSR. The relatively high loss rate of OLSR among the proactive routing protocols is caused by the higher routing overhead compared to PSR and DSDV. When the nodes are neither too sparse, so that the network connectivity is good, nor too dense, so that the channel can be spatially reused, these protocols have a fairly high PDR of over 70% for PSR, DSDV, and

DSR, and 60% to 70% for OLSR.

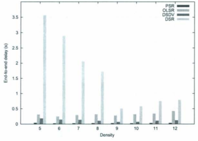


Figure 6.8: End-to-end delay in UDP vs. density

When we turn to end-to-end delay (Figure 6.8), there is a noticeable difference between DSR and the proactive protocols. In particular, DSR as reactive protocol has a rather large delay in sparse networks. This is because the long, vulnerable routes discovered during the search procedure break frequently, forcing nodes to hold packets back for an extended period before new routes are identified. Oppositely, the network sparsity does not affect proactive protocols as much because their periodic routing information exchange makes them more prepared for network structure alteration. While the delay of DSR is off the chart, that of PSR is always less than 0.05s which is also much less than that for DSDV and OLSR (0.1s to 0.43s). On a related note, the delay jitter (Figure 6.9) of PSR is significantly lower than the other three. Note that Voice-Over-IP (VoIP) applications usually discard packets that arrive too late. Therefore, the jitter among the packets actually used by the VoIP receiving agent is much smaller. Nevertheless, our metric still reflects how consistent these protocols are in delivering best-effort packets.

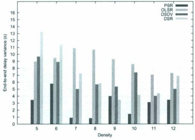


Figure 6.9: End-to-end delay jitter in UDP vs. density

6.1.5 UDP with Velocity

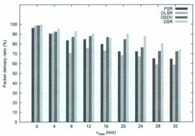


Figure 6.10: PDR in UDP vs. velocity

The same measurements are taken to test these protocols in response to different rates of node velocity. As with the case of the previous subsection, we pick three node pairs out of the 50 nodes and give them two-way CBR streams. For the entire series of different velocity caps $v_{\max} = 0, 4, 8, 12, \ldots, 32\text{m/s}$, the node density is again set to around 7 neighbors per node.

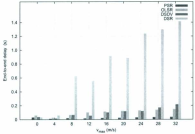


Figure 6.11: End-to-end delay in UDP vs. velocity

From the plot of PDR (Figure 6.10), we observe that DSR is able to support three voice streams with low packet loss. Specifically, the PDR of DSR, PSR, and DSDV is always over 70% even when $v_{\max} = 32$. The reliability of OLSR is relatively lower, which can go below 60% at high speed ($v_{\max} = 28$ or 32m/s). Note that all four protocols are very reliable in data delivery when $v_{\max} = 0$ or 4m/s, where the loss rates are well below 10%. Their performance in terms of PDR degrades gracefully as the rate of node velocity increases.

The end-to-end delay (Figure 6.11) presents a rather distinct landscape. In particular, the number for DSR is significantly higher than the other protocols except in low-mobility networks with $v_{\max} = 0$ or 4m/s. In all cases, the delay for PSR is much smaller compared to OLSR and DSDV. On the other hand, the measured delay jitter (Figure 6.12) indicates that all protocols become less consistent when nodes move faster. Relatively speaking, however, the variance of PSR is much smaller than the other three.

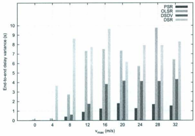


Figure 6.12: End-to-end delay jitter in UDP vs. velocity

6.2 Effectiveness Study of Small-scale Retransmission

In this section, we study the effectiveness of small-scale retransmission in particular by running computer simulation using Network Simulator ns-2. To investigate the effectiveness of small-scale retransmission, we test CORMAN with small-scale retransmission enabled and CORMAN with small-scale retransmission disabled separately. The performance improvement and explanations of the simulation results are explained in the rest of this section.

6.2.1 Experiment Settings

The channel propagation model used in ns-2 had been predominantly the Two-Ray Ground Reflection model early on. However, this model is realized to be a simplified path loss model without considering fading. In our work, we choose the Nakagami propagation model to test CORMAN in a more realistic fading environment. The probability density function of Nakagami distribution of the received signal's amplitude $X = r$ $(r \geq 0)$ is defined as:

$$f(r;\, \mu, \omega) = \frac{2\mu^{\mu}}{\Gamma(\mu)\omega^{\mu}} r^{2\mu-1} \exp\left(-\frac{\mu}{\omega} r^2\right), \tag{6.1}$$

where $\mu = \frac{\mathrm{E}^2[X^2]}{\mathrm{Var}[X^2]}$ and $\omega = \mathrm{E}\left[X^2\right]$. In ns-2, when a node has received a packet, it first calculates the received power using path loss based on the Frii Free-Space model. This value is compensated with Nakagami's fluctuation before further processing. We configure the nominal data rate at the 802.11 links to 1 Mbps.

In modeling node motion, we also adopt the random waypoint model (6.1.1) to generate the simulation scenarios.

We inject CBR (constant bit rate) data flows in the network, which are carried by UDP. Specifically, a source node generates 50 packets every second, each has a payload of 1000 bytes. This translates to a traffic rate of 400 kbps injected by a node. When comparing different CORMAN versions which has and has not the small-scale retransmission functionality, we record the packet delivery ratio (PDR), *i.e.* the fraction of packets received by the destination out of all the packets injected, and end-to-end delay average and variance. We observe that the CORMAN with small-scale retransmission enabled outperforms the CORMAN with small-scale retransmission disabled in terms of all of these metrics.

6.2.2 Performance versus Network Dimension

We make performance comparison of CORMAN under different configurations, one is with the small-scale retransmission enabled and the other is with this module disabled, and our tests are against the varying of the network density. In particular, we have network tomographies of $l \times l$ (m^2), where $l = 450, 500, 550, \ldots, 950$. We deploy

50 nodes in each of these network dimensions to test the protocols with differing node densities, and every node moves randomly with waypoint model at $v_{\max} = 20$ m/s. For each dimension scenario, we test performances of CORMAN with small-scale retransmission enabled and CORMAN with small-scale retransmission disabled in transporting CBR data flows between a randomly selected source-destination pair. We repeat this process 20 times for a given scenario. We measure the PDR, end-to-end delay, and delay jitter for both protocols and average them over the 20 repetitions of each scenario, as plotted in Figures 6.13, 6.14, and 6.15, respectively.

We observe that when the network density is relatively low, CORMAN with small-scale retransmission disabled has better PDR. In particular, when the side length of the square network boundary grows from 450m to 500m, the CORMAN with disable small-scale retransmission outperform the CORMAN with the proposed scheme enabled. When the side length of the square boundary keeps on increasing over 500m, the CORMAN with small-scale retransmission enabled outperform its opposite, and the greater the side length, the more obvious PDR gain could be achieved.

To investigate the reason behind such a phenomenon, we studied the simulation trace files and plotted out the number of packets forwarded by local relay nodes and the collisions on listed forwarders caused by local relay nodes in Figure 6.16. In particular, the left scale of Figure 6.16 gives us the number of packets forwarded by small-scale retransmitters and the collisions on listed forwarders caused by small-scale retransmitters, and the right scale gives us the ratio of these two values. It is obvious that both the small-scale forwarding and the collisions they caused increase when the network dimension goes up. Furthermore, the forwarding count increases faster than the collisions they caused. In our further investigation, we found the forwarding increases because the hops from source to destination node increase with the network dimensions. However, we found the increase of collisions is not due to the growth of hop count because we have granted nodes the priority to transmit the packets in the same batch [9]. The real reason is that a route with more hops from source to destination reduces the size of fragment transmitted by intermediate node, so the entire fragment has higher probability to be lost on both listed forwarders and local relay nodes. As the result, nodes can not predict exactly the time to start their transmission, and that is the real reason for more collisions in the network. Therefore, the collision will not grow directly against the increase of network dimensions and with a lower growth speed than that the increasing of forwarding times.

In a summary, because wireless links in dense network have relatively high reliability with relatively less hop count, and because the nodes have to try to deliver a same packet several times before discarding them, so the contribution of small-scale retransmission in dense network is in fact limited. When the density of the network decreases, the benefit achieved by using small-scale retransmission shows its advantage.

From Figure 6.14 we can see that the packet end-to-end delay in both tested schemes with and without small-scale retransmission, are nearly the same in a dense network. The scheme with local cooperative relay enabled has shorter packet end-to-end delay in a sparse network, because the local cooperative relay will save packets and avoid them to be delivered from original source again. Figure 6.15 indicates that both the CORMANs with and without small-scale retransmission have nearly the same packet end-to-end delay jitter.

6.2.3 Performance versus Velocity

We also study CORMAN's performance with the small-scale retransmission enabled or not by varying the node velocity. We conduct another set of tests in a network of 50 nodes deployed in a 700×700 (m^2) space with a varying $v_{\max}$, where $v_{\max} = 0, 3, 6, \ldots, 30$ (m/s). For each velocity scenario, we test CORMAN with small-

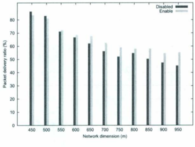


Figure 6.13: PDR vs. network dimension

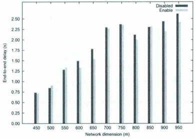


Figure 6.14: Packet delay vs. network dimension

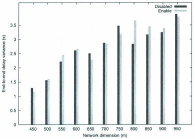


Figure 6.15: Delay jitter vs. network dimension

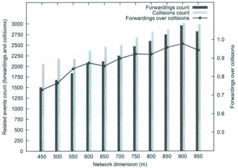


Figure 6.16: Reasons analysis of PDR reversion

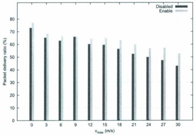


Figure 6.17: PDR vs. node velocity

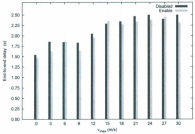


Figure 6.18: Packet delay vs. node velocity

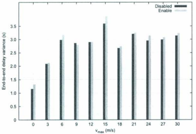


Figure 6.19: Delay jitter vs. node velocity

scale retransmission enabled and disabled in transporting CBR data flows between a randomly selected source-destination pair as well. We repeat this process 20 times for a given scenario. We collect the PDR, end-to-end delay, and delay jitter for both protocols averaged over each scenario, and plot them in Figures 6.17, 6.18, and 6.19, respectively.

From Figure 6.17, we can obviously find that for all velocity scenario, the small-scale retransmission enabled scheme outperform the small-scale retransmission disabled one because of the same reasons as we analyzed in Section 6.2.2, and so we will not explain that again here. What interesting is that even thought both the PDRs in two test conditions decrease together when nodes move faster, the PDR gain by using small-scale retransmission in CORMAN grows when the velocity goes up. That is due to the scoring function in the small-scale retransmitter is a real time and distributed algorithm, and it can provide the best retransmitter on time to reduce the effect of nodes mobility. As the result, the small-scale retransmission has more effectiveness to fix up the break links when the node velocities go up.

Also, CORMAN with small-scale retransmission enabled has shorter end-to-end delay and almost same end-to-end delay jitter compare to that with it disabled, and that is because the same reasons we analyzed in Section 6.2.2.

6.3 Overall Performance of CORMAN

In this section, we study the performance of CORMAN by running computer simulation using Network Simulator ns-2 (version 2.34). We compare it against AODV with varying network densities and node mobility rates. The performance improvement and explanations of these results are explained in the rest of this section.

6.3.1 Experiment Settings

The basic configurations of the simulations for the overall performance in CORMAN are nearly the same as those in the simulations for the effectiveness of small-scale retransmission in Section 6.2, where we choose the Nakagami propagation model to test CORMAN's overall performance, configure the nominal data rate at the 802.11 links to 1 Mbps, and adopt the random waypoint mobility model to generate the simulation scenarios. As well, we inject CBR data flows in the network carried by UDP with the same data rate as we specified in Section 6.2. We compare the overall performance of CORMAN with that of AODV [6]. We select AODV as the baseline because AODV is a widely adopted routing protocol in MANETs, and its behavior both in ns-2 and real network operations is well understood by the research community.

6.3.2 Performance versus Network Dimension

We first compare the performance of CORMAN and AODV with different network dimensions. Specifically, we have network tomographies of $l \times l$ (m^2), where $l = 250, 300, 350, \ldots, 1000$. We deploy 50 nodes in each of these network dimensions to test the protocols with differing node densities. These nodes move following the random waypoint model with $v_{\max} = 10$ m/s. For each dimension scenario, we test CORMAN and AODV's capabilities in transporting CBR data flows between a randomly selected source-destination pair. We repeat this process 5 times for a given scenario. We measure the PDR, end-to-end delay, and delay jitter for both protocols and average them over the 5 repetitions of each scenario, as plotted in Figures 6.20, 6.21, and 6.22, respectively.

We observe that CORMAN has a PDR (Figure 6.20) of about 95% for dense networks (*i.e.*, $250 \leq l \leq 500$ m). As the node density decreases, this rate gradually goes down to about 60%. In contrast, AODV's PDR ranges between 60% and 80% for dense networks and quickly drops to around 20% for sparse networks. (We use a red plotting series to indicate the relative performance of CORMAN over PDR in all of our figures.) There are two reasons for the PDR penalty for AODV to operate in sparse networks. First, data packets are forwarded using traditional IP forwarding in AODV. When channel quality varies (as emulated by the Nakagami model), a packet may be lost at the link layer. After a few failed retransmits, it will be dropped by the network layer. CORMAN, however, is designed to utilize such link effects so that at least one downstream node would be available despite the link variation. CORMAN facilitates opportunistic data forwarding using the link quality diversity at different receivers and allows them to cooperate with each other with a minimal overhead. Consequently, CORMAN has a strong resilience to link quality fluctuation and node mobility. Second, the route search of AODV does not function well with unreliable links. Recall that, in AODV, when a node finds that it does not have a next hop available for a given data packet, it broadcasts a RREQ (route request) to find one. Both the destination and any intermediate node that has a valid cached route can reply with a RREP (route reply). When links were perfectly symmetric, the RREP packet would take the inverse path leading to the initiating node of the route search. However, when links suffer from fading, the RREP packets may not be able to propagate back to the initiator because of transient low link quality in the reverse direction. As a result, it takes AODV much longer to obtain stable routes. In fact, this performance loss has been observed in earlier studies of AODV when large numbers of unidirectional links are present in the network [47]. In our ns-2 simulation using the Nakagami propagation model, the independent link quality fluctuation in both directions essentially produces temporary unidirectional links. This negative effect of fading links on AODV can be mediated to a degree when the node density increases. Therefore, in our next set of tests (Section 6.3.3), all simulation scenarios have a fairly higher node density for AODV to function reasonably well.

We are also interested in the end-to-end delay and its variance of CORMAN and AODV. Figure 6.21 presents the end-to-end delay of these protocols in different dimension scenarios. We see that, when the node density is higher (*i.e.*, $250 \leq l \leq 500$ m), CORMAN has a shorter delay than AODV. For sparser scenarios (*i.e.*, $550 \leq l \leq 1000$ m), CORMAN's delay is slightly longer than AODV but comparable. In CORMAN's implementation, a node determines that it can no longer contribute to a batch's progression if it has not seen an update of the map after 10 retransmits of its fragment. This is similar to 10 retries at the link layer in IP style forwarding. (The default retry limit in 802.11 is 7.) Thus, the not-so-short end-to-end delay of CORMAN is caused by the larger number of data retransmits. This is a relatively small cost to pay for a significantly higher PDR (Figure 6.20). The delay jitter (standard deviation) measured for CORMAN and AODV has a similar relative performance for these scenarios as in Figure 6.22. This is also because of the larger retry limit of CORMAN (10 times) compared to AODV over 802.11 (7 times).

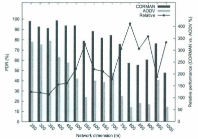


Figure 6.20: PDR vs. network dimension

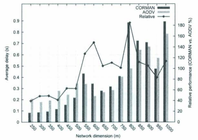


Figure 6.21: Packet delay vs. network dimension

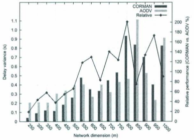


Figure 6.22: Delay jitter vs. network dimension

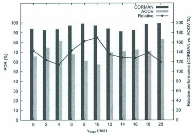


Figure 6.23: PDR vs. node velocity

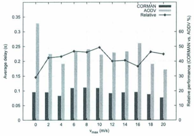


Figure 6.24: Packet delay vs. node velocity

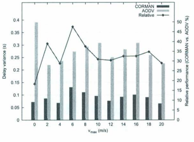


Figure 6.25: Delay jitter vs. node velocity

6.3.3 Performance versus Velocity

We also study CORMAN's performance and compare it to AODV in different rates of node velocity. We conduct another set of tests in a network of 100 nodes deployed in a 300×300 (m^2) space with a varying $v_{\max}$, where $v_{\max} = 0, 2, 4, 6, \ldots, 20$ (m/s). For each velocity scenario, we test CORMAN and AODV's performance in transporting CBR data flows again between a randomly selected source-destination pair. We repeat this process 5 times for a given scenario. We collect the PDR, end-to-end delay, and delay jitter for both protocols averaged over each scenario, and plot them in Figures 6.23, 6.24, and 6.25, respectively. Note that these are fairly dense networks so that AODV has a reasonably high PDR. Since the network diameter is rather small in this case, the measurements are fairly consistent across these different velocity scenarios.

From Figure 6.23, we observe that CORMAN's PDR is constantly around 95% while that of AODV varies between 57% and 82%. With this very high network density, AODV's route search succeeds in majority of the cases. Yet, it still does not have the same level of robustness against link quality changes as CORMAN. Compared to AODV, CORMAN has only a fraction of the end-to-end delay and variance (Figures 6.24 and 6.25) for two reasons. First, the opportunistic data forwarding scheme in CORMAN allows some packets to reach the destination in fewer hops than AODV. Second, the proactive routing (PSR) in CORMAN maintains full-on route information, whereas AODV still has to search for them if a route is broken. Although route search in AODV usually succeeds in dense networks with fluctuating link quality, the delay introduced by this process is inevitable.

Based on these observations, we conclude that CORMAN can maintain its performance despite high rate of node velocity with realistic channels emulated by the Nakagami model. Therefore, it is very suitable for many mobile ad hoc network applications, *e.g.*, Vehicular Ad-hoc Network (VANET) applications.

6.4 Summary

In this chapter, we test the proposed schemes with Network Simulator 2 to study their particular and overall performances. Specifically, the overhead in PSR is only a small fraction or a magnitude smaller than that in other three baselines, and meanwhile PSR can maintain a better performance on TCP throughput, packet end-to-end delay, and end-to-end delay jitter compared to other protocols. The particular evaluation for small-scale retransmission shows us the small-scale retransmission can effectively bridge broken links in sparse and high mobility networks, and it can provide us up to 15% performance gain in Packet Delivery Ratio (PDR) and a little improvement in packet end-to-end delay. When we test CORMAN as an entire system, we found CORMAN has the PDR up to 4 times of the PDR in AODV. Hence, we proved that the systematic solutions we proposed to implement opportunistic data forwarding in mobile ad hoc networks works quite well.

Chapter 7

Conclusions, Discussions, and Future Work

7.1 Conclusions

In this thesis, we have proposed CORMAN as an opportunistic routing scheme for mobile ad hoc networks. CORMAN is composed of three components. 1) PSR — a proactive source routing protocol, 2) large-scale live update of forwarder list, and 3) small-scale retransmission of missing packets. All of these explicitly utilize the broadcasting nature of wireless channels and are achieved via efficient cooperation among participating nodes in the network. Essentially, when packets of the same flow are forwarded, they can take different paths to the destination. For example, in Figure 7.1, the route between nodes X and Y as determined by the routing module is indicated by the yellow band. The solid circles represent the listed forwarders and the hollow ones are the small-scale retransmitters. In actual operations of CORMAN, packets p_1, p_2, and p_3 can take separate routes around this band depending on the transient link quality in the network. Such a decision is made on a per-hop and per-

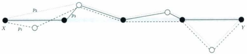


Figure 7.1: Packet trajectories

packet basis. Through computer simulation, CORMAN is shown to have superior performance measured in PDR, delay, and delay jitter.

7.2 Discussions

7.2.1 Proactive Source Routing Related

In particular, the PSR is motivated by the need of supporting opportunistic data forwarding in *mobile* ad hoc networks. In order generalize the milestone work of ExOR for it to function in such networks, we needed a proactive source routing protocol. Such a protocol should provide more topology information than just distance vectors but has a significantly smaller overhead than link-state routing protocols; even the MPR technique in OLSR would not suffice this need. Thus, we put forward a tree-based routing protocol, PSR, inspired by PFA and WRP. Its routing overhead per time unit per node is in the order of the number of the nodes in the network as with DSDV, but each node has the full-path information to reach all other nodes. For it to have a very small footprint, PSR's route messaging is designed to be very concise. First, it uses only one type of message, *i.e.*, the periodic route update, both to exchange routing information and as hello beacon messages. Second, rather than packaging a set of discrete tree edges in the routing messages, we package a converted binary tree to reduce the size of the payload by about a half. Third, we interleave full dump messages with differential updates so that, in relatively stable networks, the differential updates are much shorter than the full dump messages. To further reduce the size of the differential updates, when a node maintains its routing tree as the network changes, it tries to minimize the alteration of the tree. As a result, the routing overhead of PSR is only a fraction or less compared to DSDV, OLSR, and DSR as evidenced by our experiments. Yet, it still has similar or better performance in transporting TCP and UDP data flows in mobile networks of different velocity rates and densities.

In the simulation in this work, we used PSR to support traditional IP forwarding for a closer comparison with DSDV and OLSR, while DSR still carried source routed messages. In our simultaneous work, CORMAN [37], we tested PSR's capability in transporting source routed packets for opportunistic data forwarding, where we also found that PSR's small overhead met our initial goal. That being said, as indicated in Section 6.1.2 earlier, while alleviating forwarding nodes from table lookup, DSR's source routing is especially vulnerable in rapidly changing networks. The reason is that, as a source routed packet progresses further from its source, the path carried by the packet can become obsolete, forcing an intermediate node that cannot find the next hop of the path to drop the packet. This is fundamentally different from traditional IP forwarding in proactive routing with more built-in adaptivity, where the routing information maintained at nodes closer to the destination is often more updated than the source node. Although out of the scope of this research, it would be an interesting exploration to allow intermediate nodes running DSR to modify the path carried by a source routed packet for it to use its more updated knowledge to route data to the destination. This is in fact exactly what PSR does when we used it to carry source routed data in CORMAN. Granted, this opens up an array of security issues, which themselves are part of a vast research area.

As with many protocol designs, in many situations working on PSR, we faced tradeoffs of sorts. Striking such balances not only gave us the opportunity to think about our design twice, but also made us understand the problem at hand better. One particular example is related to trading computational power for data transfer performance. During one route exchange interval, a node receives a number of routing messages from its neighbors. It needs to incorporate the updated information to its knowledge base and share it with its neighbors. The question is when should these two events happen. Although incorporating multiple trees at one time is computationally more efficient, we chose to do that immediately after receiving an update from a neighbor. As such, the more accurate information takes effect without any delay. Otherwise, when a data packet is forwarded to a neighbor that no longer exists, it causes link layer retrial, backlogging of subsequent packets, and TCP congestion avoidance and retransmission. With the broadcast and shared nature of the wireless channel, the effects above are adversary to all other data flows in the area. Therefore, in research on multi-hop wireless networking, it almost always makes sense for us to minimize any impact on the network's communication resources even if there is penalty in other aspects. When it comes to when a node should share its updated route information with its neighbors, we chose to delay it until the end of the cycle so that only one update is broadcast in each period. If a node were to transmit it immediately when there is any change to its routing tree, it would trigger an explosive chain reaction and the network would be overwhelmed by the route updates. As we found out in our preliminary tests, this is the primary reason that WRP's overhead was significantly higher than the other protocols under study. PSR has just opened the box for us, and there will be many more things that we would like to investigate about it.

7.2.2 About Large-scale Live Update and Small-scale Retransmission

The large-scale live update is another way to utilize the broadcast nature in wireless communication network, especially for the opportunistic data forwarding with proactive routing protocols in MANETs. In particular, to avoid too much routing overhead that is injected into network, the proactive routing protocols can not update the topology changes in a event driven manner but timer driven, which means the routing update should propagate from destination to source in periodical way as we did in PSR (Chapter III). As a result, the further from the source node to destination node, the less accurate routing information maintained on the source node. Coherently, the Route-Prioritized Contention based opportunistic data forwarding (Section 2.1.1.2) always wants the intermediate node that is nearest to the destination to forward the packets first, where much fresher routing information is maintained than upstream nodes which will forward rest packets later. Hence, large-scale live update uses the broadcast nature further in wireless opportunistic forwarding network, and when a node take itself as a frontier (Section 4.2) of the batch, it will update the forwarder list and pack the new one in the data packet. Therefore, when the data packets are delivered towards the destination, the fresher routing information can be propagated towards source node more quickly with no additional overhead.

Furthermore, in the small-scale retransmission we proposed in Chapter V, we broaden the opportunistic data forwarding further. In wireless communication networks, the traditional IP forwarding, opportunistic data forwarding, and opportunistic data forwarding with small-scale retransmission compose an evolution process. In particular, the most fundamental approach is the IP forwarding, which is initially proposed for the Internet and only one next hop address marks the intended receiver in forwarding process. Then, ExOR [9] makes a group of receiver to be intended by including a forwarder list in data packet, and make all receivers forward packet in a a prioritized order. Now, our small-scale retransmission not only grants the forwarding ability to the receivers included in the forwarder list, but also lets the nodes which are not shown in the forwarder list but between two of them to participate in data forwarding. The broadened opportunistic data forwarding exploits the broadcast nature one step further.

7.3 Future Work

Research based on CORMAN can be extended in the following interesting ways.

1. It would be informative to further test CORMAN. For example, we can compare CORMAN to ExOR and IP forwarding in static multi-hop networks with varying link quality to study their relative capabilities in data transfer. We will also test these data transfer techniques with multiple simultaneous flows present to study how well they share the network resources. Through these tests, we will be able further optimize some parameters of CORMAN, such as the retry limit.

2. The coordination among multiple qualified small-scale retransmitters can be achieved with better measures than RSSI. In particular, if the "suitability" score is based on transient link quality rather than historic information, we will be able better utilize the receiver diversity. Apparently, this would require non-trivial coordination among these retransmitters, which could be challenging especially when we aim for zero extra overhead.

3. Nodes running CORMAN forward data packets in fragments. When the source

and destination nodes are separated by many hops, it should allow nodes at different segment of the route to operate simultaneously. That is, a pipeline of data transportation could be achieved by better spatial channel reuse. The design of CORMAN can be further improved to address this explicitly. This may involve timing node back off more precisely and tightly, or even devising a completely different coordination scheme.

The potential of cooperative communication in multi-hop wireless networks is yet to be unleashed at higher layers, and CORMAN is only an example. PSR has just opened the box for us, and there will be many more things that we would like to investigate about it.

Bibliography

[1] I. Chlamtac, M. Conti, and J.-N. Liu, "Mobile Ad hoc Networking: Imperatives and Challenges," *Ad Hoc Networks*, vol. 1, no. 1, pp. 13–64, July 2003.

[2] R. Rajaraman, "Topology control and routing in ad hoc networks: A survey," *SIGACT News*, vol. 33, pp. 60–73, June 2002.

[3] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *Computer Communication Review*, pp. 234–244, October 1994.

[4] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," *RFC 3626*, October 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3626.txt

[5] D. B. Johnson, Y.-C. Hu, and D. A. Maltz, "On The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4," *RFC 4728*, February 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4728.txt

[6] C. E. Perkins and E. M. Royer, "Ad hoc On-Demand Distance Vector (AODV) Routing," *RFC 3561*, July 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3561.txt

[7] T. M. Cover and A. A. E. Gamal, "Capacity Theorems for the Relay Channel," *IEEE Transactions on Information Theory*, vol. 25, no. 5, pp. 572–584, September 1979.

[8] A. Nosratinia, T. E. Hunter, and A. Hedayat, "Cooperative Communication in Wireless Networks," *IEEE Communications Magazine*, vol. 42, no. 10, pp. 74–80, October 2004.

[9] S. Biswas and R. Morris, "ExOR: Opportunistic Multi-Hop Routing for Wireless Networks," in *Proceedings of ACM Conference of the Special Interest Group on Data Communication (SIGCOMM)*, Philadelphia, PA, USA, August 2005, pp. 133–144.

[10] P. Larsson, "Selection Diversity Forwarding in a Multihop Packet Radio Network With Fading Channel and Capture," *ACM Mobile Computing and Communications Review*, vol. 5, no. 4, pp. 47–54, October 2001.

[11] E. Rozner, J. Seshadri, Y. Mehta, and L. Qiu, "Simple Opportunistic Routing Protocol for Wireless Mesh Networks," in *Proceedings of the 2nd IEEE Workshop on Wireless Mesh Networks (WiMesh)*, Sep. 2006, pp. 48–54.

[12] M. Kurth, A. Zubow, and J.-P. Redlich, "Cooperative Opportunistic Routing Using Transmit Diversity in Wireless Mesh Networks," in *Proceedings of the 27th IEEE International Conference on Computer Communication (INFOCOM)*, Apr. 2008, pp. 1310–1318.

[13] M. Naghshvar and T. Javidi, "Opportunistic Routing with Congestion Diversity in Wireless Multi-hop Networks," in *Proceedings of the 29th IEEE International Conference on Computer Communication (INFOCOM)*. Piscataway, NJ, USA: IEEE Press, 2010, pp. 496–500.

[14] K. Zeng, Z. Yang, and W. Lou, "Opportunistic Routing in Multi-Radio Multi-Channel Multi-Hop Wireless Networks," in *Proceedings of the 29th IEEE International Conference on Computer Communication (INFOCOM)*. Piscataway, NJ, USA: IEEE Press, 2010, pp. 476–480.

[15] S. Yang, F. Zhong, C. K. Yeo, B. S. Lee, and J. Boleng, "Position Based Opportunistic Routing for Robust Data Delivery in MANETs," in *Proceedings of the 2009 IEEE Conference on Global Telecommunications (GLOBECOM)*, Honolulu, Hawaii, USA, December 2009, pp. 1325–1330.

[16] Z. Zhong and S. Nelakuditit, "On the Efficacy of Opportunistic Routing," in *Proceedings of the 4th IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, Jun. 2007, pp. 441–450.

[17] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading Structure for Randomness in Wireless Opportunistic Routing," in *Proceedings of ACM Conference of the Special Interest Group on Data Communication (SIGCOMM)*, Kyoto, Japan, August 2007, pp. 169–180.

[18] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom)*, San Diego, CA, USA, 2003, pp. 134–146.

[19] J. Ma, Q. Zhang, C. Qian, and L. M. Ni, "Energy-Efficient Opportunistic Topology Control in Wireless Sensor Networks," in *Proceedings of the 1st International MobiSys Workshop on Mobile Opportunistic Networking (MobiOpp)*. New York, NY, USA: ACM, 2007, pp. 33–38.

[20] I. Leontiadis and C. Mascolo, "GeOpps: Geographical Opportunistic Routing for Vehicular Networks," in *Proceedings of the IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM)*, Helsinki, Finland, June 2007, pp. 1–6.

[21] B. Radunović, C. Gkantsidis, P. Key, and P. Rodriguez, "An Optimization Framework for Opportunistic Multipath Routing in Wireless Mesh Networks," in *Proceedings of the 27th IEEE International Conference on Computer Communication (INFOCOM)*, Apr. 2008, pp. 2252–2260.

[22] D. Koutsonikolas, C.-C. Wang, and Y. C. Hu, "CCACK: Efficient Network Coding Based Opportunistic Routing through Cumulative Coded Acknowledgments," in *Proceedings of the 29th IEEE International Conference on Computer Communication (INFOCOM)*. Piscataway, NJ, USA: IEEE Press, 2010, pp. 2919–2927.

[23] R. C. Shah, S. Wiethölter, A. Wolisz, and J. M. Rabaey, "When does Opportunistic Routing Make Sense?" in *Proceedings of the 3rd Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, Mar. 2005, pp. 350–356.

[24] J. Kim and S. Bohacek, "A Comparison of Opportunistic and Deterministic Forwarding in Mobile Multihop Wireless Networks," in *Proceedings of the 1st International MobiSys Workshop on Mobile Opportunistic Networking (MobiOpp)*. New York, NY, USA: ACM, 2007, pp. 9–16.

[25] K. Zeng, W. Lou, and H. Zhai, "On End-to-End Throughput of Opportunistic Routing in Multirate and Multihop Wireless Networks," in *Proceedings of the 27th IEEE International Conference on Computer Communication (INFOCOM)*, Apr. 2008, pp. 816–824.

[26] L. Cerdà-Alabern, V. Pla, and A. Darehshoorzadeh, "On the Performance Modeling of Opportunistic Routing," in *Proceedings of the 2nd International Workshop on Mobile Opportunistic Networking (MobiOpp)*. New York, NY, USA: ACM, 2010, pp. 15–21.

[27] M.-H. Lu, P. Steenkiste, and T. Chen, "Video Transmission over Wireless Multihop Networks Using Opportunistic Routing," in *Proceedings of the 16th IEEE International Packet Video Workshop (PV)*, Nov. 2007, pp. 52–61.

[28] F. Wu, T. Chen, S. Zhong, L. E. Li, and Y. R. Yang, "Incentive-Compatible Opportunistic Routing for Wireless Networks," in *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking (MobiCom)*. New York, NY, USA: ACM, 2008, pp. 303–314.

[29] J. J. Garcia-Luna-Aceves and S. Murthy, "A Path-Finding Algorithm for Loop-Free Routing," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 148–160, February 1997.

[30] J. Behrens and J. J. Garcia-Luna-Aceves, "Distributed, Scalable Routing based on Link-State Vectors," in *Proceedings of ACM SIGCOMM*, 1994, pp. 136–147.

[31] K. Levchenko, G. M. Voelker, R. Paturi, and S. Savage, "XL: An Efficient Network Routing Algorithm," in *Proceedings of ACM SIGCOMM*, 2008, pp. 15–26.

[32] S. Murthy and J. J. Garcia-Luna-Aceves, "An Efficient Routing Protocol for Wireless Networks," *Mobile Networks and Applications*, vol. 1, no. 2, pp. 183–197, October 1996.

[33] J. J. Garcia-Luna-Aceves and M. Spohn, "Source-Tree Routing in Wireless Networks," in *Proceedings of the 7th Annual International Conference on Network Protocols (ICNP'99)*, Toronto, Canada, October 1999, pp. 273–282.

[34] X. Yu, "Distributed Cache Updating for the Dynamic Source Routing Protocol," *IEEE Transactions on Mobile Computing*, vol. 5, no. 6, pp. 609–626, June 2006.

[35] Y.-C. Hu and D. B. Johnson, "Implicit Source Routes for On-Demand Ad Hoc Network Routing," in *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'01)*, Long Beach, CA, USA, October 2001, pp. 1–10.

[36] B. Hu and H. Gharavi, "DSR-Based Directional Routing Protocol for Ad Hoc Networks," in *Proceedings of the 2007 IEEE Conference on Global Telecommunications (GLOBECOM)*, Washington D.C., DC USA, November 2007, pp. 4936–4940.

[37] Z. Wang, Y. Chen, and C. Li, "CORMAN: a Novel Cooperative Opportunistic Routing Scheme in Mobile Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications*, to appear in 2012.

[38] Z. Wang, C. Li, and Y. Chen, "PSR: Proactive Source Routing in Mobile Ad Hoc Networks," in *Proceedings of the 2011 IEEE Conference on Global Telecommunications (GLOBECOM)*, Houston, TX USA, December 2011.

[39] Z. Wang, Y. Chen, and C. Li, "A New Loop-Free Proactive Source Routing Scheme for Opportunistic Data Forwarding in Wireless Networks," *IEEE Communications Letters*, to appear in 2012.

[40] ——, "PSR: A Light-Weight Proactive Source Routing Protocol for Mobile Ad Hoc Networks," in *2012 IEEE International Conference on Computer Communications (INFOCOM)*, waiting for result.

[41] D. West, *Introduction to Graph Theory (2nd Edition)*. Upper Saddle River, NJ, USA: Prentice Hall, August 2000.

[42] Z. Wang, C. Li, and Y. Chen, "Local Cooperative Retransmission in Opportunistic Data Forwarding," in *2012 IEEE International Conference on Communications (ICC)*, waiting for result.

[43] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Measurement-Based Models of Delivery and Interference in Static Wireless Networks," in *Proceedings of ACM SIGCOMM*, August 2006, pp. 51–62.

[44] M.-H. Lu, P. Steenkiste, and T. Chen, "Design, Implementation and Evaluation of an Efficient Opportunistic Retransmission Protocol," in *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking (MobiCom)*. New York, NY, USA: ACM, 2009, pp. 73–84.

[45] J. Bardwell, "Converting Signal Strength Percentage to dBm Values," WildPackets, November 2002. [Online]. Available: http://www.wildpackets.com/elements/whitepapers/Converting_Signal_Strength.pdf

[46] U. Researchers at UC Berkeley, LBL and X. PARC, "ns manual," May 2010. [Online]. Available: http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf

[47] M. K. Marina and S. R. Das, "Routing Performance in the Presence of Unidirectional Links in Multihop Wireless Networks," in *The Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02)*, Lausanne, Switzerland, June 2002, pp. 12–23.