ROUTING IN MOBILE AD-HOC NETWORKS: SCALABILITY AND EFFICIENCY

Rendong Bai
University of Kentucky, rdbai@cs.uky.edu

ABSTRACT OF DISSERTATION

Rendong Bai

The Graduate School

University of Kentucky

2007

ROUTING IN MOBILE AD-HOC NETWORKS: SCALABILITY AND EFFICIENCY

ABSTRACT OF DISSERTATION

A dissertation submitted in partial fulfillment of the
requirements of the degree of Doctor of Philosophy in the
College of Engineering at the University of Kentucky

By

Rendong Bai

Lexington, Kentucky

Director: Dr. Mukesh Singhal, Department of Computer Science

Lexington, Kentucky

2007

ABSTRACT OF DISSERTATION

ROUTING IN MOBILE AD-HOC NETWORKS: SCALABILITY AND EFFICIENCY

Mobile Ad-hoc Networks (MANETs) have received considerable research interest in recent years. Because of dynamic topology and limited resources, it is challenging to design routing protocols for MANETs. In this dissertation, we focus on the scalability and efficiency problems in designing routing protocols for MANETs.

We design the Way Point Routing (WPR) model for medium to large networks. WPR selects a number of nodes on a route as waypoints and divides the route into segments at the waypoints. Waypoint nodes run a high-level inter-segment routing protocol, and nodes on each segment run a low-level intra-segment routing protocol. We use DSR and AODV as the inter-segment and the intra-segment routing protocols, respectively. We term this instantiation the DSR Over AODV (DOA) routing protocol.

We develop Salvaging Route Reply (SRR) to salvage undeliverable route reply (RREP) messages. We propose two SRR schemes: SRR1 and SRR2. In SRR1, a salvor actively broadcasts a one-hop salvage request to find an alternative path to the source. In SRR2, nodes passively learn an alternative path from duplicate route request (RREQ) packets. A salvor uses the alternative path to forward a RREP when the original path is broken.

We propose Multiple-Target Route Discovery (MTRD) to aggregate multiple route requests into one RREQ message and to discover multiple targets simultaneously. When a source initiates a route discovery, it first tries to attach its request to existing RREQ packets that it relays. MTRD improves routing performance by reducing the number of regular route discoveries.

We develop a new scheme called Bilateral Route Discovery (BRD), in which both source and destination actively participate in a route discovery process. BRD consists of two halves: a *source route discovery* and a *destination route discovery*, each searching for the other. BRD has the potential to reduce control overhead by one half.

We propose an efficient and generalized approach called Accumulated Path Metric (APM) to support High-Throughput Metrics (HTMs). APM finds the shortest path without collecting topology information and without running a shortest-path algorithm. Moreover, we develop the Broadcast Ordering (BO) technique to suppress unnecessary RREQ transmissions.

KEYWORDS: Mobile ad-hoc networks, routing protocols, scalability, efficiency, performance evaluation

Rendong Bai

January 08, 2008

ROUTING IN MOBILE AD-HOC NETWORKS: SCALABILITY AND EFFICIENCY

By

Rendong Bai

Dr. Mukesh Singhal

Director of Dissertation

Dr. Raphael Finkel

Director of Graduate Studies

January 11, 2008

DISSERTATION

Rendong Bai

The Graduate School

University of Kentucky

2007

ROUTING IN MOBILE AD-HOC NETWORKS: SCALABILITY AND EFFICIENCY

DISSERTATION

A dissertation submitted in partial fulfillment of the requirements of the degree of Doctor of Philosophy in the College of Engineering at the University of Kentucky

By

Rendong Bai

Lexington, Kentucky

Director: Dr. Mukesh Singhal, Department of Computer Science

Lexington, Kentucky

2007

DEDICATION

To my wife Ying, my son Larry, and my parents, Hongcheng and Bing.

ACKNOWLEDGMENTS

There are many people I would like to acknowledge toward the completion of my graduate study. Foremost, thanks go to my advisor, Dr. Mukesh Singhal, for his invaluable support, guidance, and encouragement during my school years. His understanding and patient counseling have been a key force that helped me get through many difficulties and frustrations. I am very grateful to have him as my mentor; he has taught me not only to be a good scientist, but also to be a strong person in life.

Many sincere thanks to Dr. Raphael Finkel, Dr. Dakshnamoorthy Manivannan, and Dr. Cai-Cheng Lu for serving as my committee members and to Dr. J. Robert Heath for serving as the outside examiner for my dissertation. Their valuable input and scientific guidance have been critical for my research.

I would also like to thank everybody in the lab: Huaizhi Li, Venkata C. Giruka, Yongwei Wang, Yan Sun, Saikat Chakrabarti, and Santosh Chandrasekhar. I appreciate them not only for their excellent technical help, but also for being good listeners and supporters.

I am blessed to have a very supportive and loving family: my wife, Ying Liang, my son, Larry, and my parents, Mr. Hongcheng Bai and Mrs. Bing Jin. They have given me tremendous love, support and encouragement. Without their encouragement, I wouldn't have been able to achieve my goals.

It is impossible to end my acknowledgements without mentioning the graduate program in the Department of Computer Science and the many faculty and staff I have interacted with during my graduate school training at the University of Kentucky. I appreciate very much all of their support and help.

Table of Contents

List of Tables

List of Figures

List of Files

1.	Dissertation_RendongBai.pdf	1,108 kilobytes

Chapter 1

Introduction

This dissertation focuses on the scalability and the efficiency problems in designing routing protocols for mobile ad-hoc networks (MANETs). The dissertation first presents a scalable routing protocol that applies a hierarchical structure. Then, the dissertation presents three routing optimization techniques: salvaging route reply, multiple-target route discovery, and bilateral route discovery. Finally, the dissertation presents a generalized and efficient scheme to support high-throughput routing metrics.

It took telephone 75 years and television 13 years to acquire 50 million users. It took the Internet 5 years to reach the same milestone. The Internet has become the fastest growing medium. Today, more than 1,244 million people around the world use the Internet [1]. Different people may have different experiences with computers and the Internet. The following is my experience:

- 20 years ago (1987), I thought a calculator was a computer.
- 10 years ago (1997), I was using a Pentium computer, which had a LAN connection but did not connect to the Internet.
- Today (2007), I am using a dual-core computer, which connects to the Internet most of the time. The connection is either wired or wireless.

During the first ten years (1987-1997), computing evolved from centralized mode (mainframe) to distributed mode (PC). During the second ten years (1997-2007), the processing power of computers kept improving, but networking was the fundamental technology for the fast growth of the Internet. Traditionally, computers connected to a network using a cable. In the past few years, connecting to a network using wireless links became increasingly popular. What will computers and networks look like after 10 or 20 years? It is difficult to predict tomorrow's world because there are too many factors and variables. But I think the following things are likely to happen:

- In 2017, our world will look like a cyber world. Computers, embedded systems and sensors are ubiquitous. Typically, they are connected to some kind of network. For example, cars act as information platforms. They can communicate with each other for purposes such as avoiding a collision.

- In 2027, we will be able to live and work in a cyber world. Visualization and virtual environment are practical. Employees do not have to commute. They can work in a virtual office in their virtual corporate building. We can watch a basketball game at Rupp Arena even if the WildCats are playing a road game.

In these scenarios, traditional wired networks will still play an important role. For example, they will provide backbone data transmission. However, the demand for exchanging information and accessing data conveniently (e.g., from anywhere, untethered) motivates the rising and dramatic growth of wireless networks. The best examples are cellular telephone networks and wireless LANs (WLANs). Both cellular networks and WLANs require a fixed infrastructure: base stations for cellular networks and access points for WLANs. However, investment in fixed infrastructures may be too expensive or not feasible in certain scenarios, such as rural areas and battle fields. Furthermore, maintaining and upgrading these infrastructured networks are costly and time-consuming. Therefore, it is desirable to establish a network without a fixed infrastructure. The movement from infrastructured wireless networks to infrastructure-less wireless networks resembles the transition from centralized computing to distributed computing in 1990s.

1.1 Wireless Networks without Infrastructures

In an infrastructure-less wireless network, nodes operate as both routers and hosts. A destination node may be located beyond the radio range of a source node, and thus the source needs a number of intermediate nodes to relay a message to the destination. A path from the source to the destination may consist of multiple hops (wireless links). Infrastructure-less wireless networks are desirable for many applications, ranging from small, static networks to large, dynamic networks. The two most actively researched types of infrastructure-less wireless networks are MANETs and wireless sensor networks (WSNs). This dissertation focuses on MANETs. Compared to MANETs, WSNs are more resource-constrained and more data-centric. Sensors have limited size, processing power, storage, and energy. We may use WSNs in environment monitoring, health care, home automation, and so on.

A MANET is a self-organized system of mobile nodes which communicate over wireless links. Nodes may move randomly and thus the network topology may change rapidly and unpredictably. Nodes cooperatively relay each other's packets toward destinations, typically through multiple hops. The following are a number of example applications of MANETs: civilian wireless networks for campuses and conference rooms, emergency situations such as fire and disaster relief, military scenarios such as communication in battlefields, vehicle-to-vehicle communication in intelligent transportation systems, and personal area networks connecting cell phones, PDAs, laptops, and wearable computers.

Although MANETs offer many advantages, the design of network protocols for MANETs generally is more complicated than wired networks and infrastructured wireless networks. MANETs pose many challenges to network researchers, and the following is a list of key challenges:

- *Medium access control*. In MANETs, one radio channel is shared by a number of neighboring nodes. A major challenge is how to design a medium access control (MAC) protocol to coordinate the access of nodes to the shared channel. A good MAC protocol uses the limited wireless resources (e.g., bandwidth and batteries) efficiently and provides satisfactory network throughput.

- *Routing*. In MANETs, communication between two nodes typically involves multiple hops, and the network topology changes dynamically. Routes may last for a short time only and need to be re-established often. It is difficult to design an efficient routing protocol for MANETs. Routing protocols that work well in wired networks may incur too much control overhead in MANETs.

- *Security and node cooperation*. MANETs feature open wireless communication, dynamic topology, and constrained resources. These features bring new challenges to network security. A node may simply behave selfishly and may not forward packets for other nodes. An attacker can passively listen to the channel without being detected, or actively modify or inject packets to disrupt normal network operations.

- *Service and resource discovery*. In MANETs, a number of nodes may provide services or resources such as storage, authentication, location service, and Internet access. However, other nodes may have little or no knowledge about these services or resources. Therefore, we need mechanisms to help nodes discover services or resources in the network.

- *Quality of service*. QoS refers to the ability of a network to provide reliable service to selected network traffic. QoS is necessary in many multimedia applications. However, providing QoS in MANETs is difficult because link quality, traffic load, and network topology are dynamic and unpredictable.

- *Billing*. MANETs or their variations are slowly finding their way into civilian environments. This movement leads to the issue of billing, which addresses and stimulates cooperation among nodes. For example, a small number of gateway nodes in the network have Internet access. Other non-gateway nodes connect to the Internet through the gateways. We need a billing mechanism to ensure that non-gateway nodes share the cost of Internet access paid by gateways.

In addition to the above list, there are other challenges in MANETs, such as energy saving, location information management, clustering, and automatic network configuration. Among these challenges, this dissertation focuses on the routing issue in MANETs. The next section presents the problem statement.

1.2 The Problem Statement

Routing refers to the process of moving data packets from sources to destinations. Routing protocols specify how routers communicate with each other to establish routes among nodes in the network. Researchers and engineers have studied routing protocols for wired networks extensively. We can classify these protocols into different categories based on different criteria. If based on the delivery pattern, we can classify routing schemes into unicast, broadcast, multicast, and anycast. Unicast is the most common form of message delivery on the Internet. If based on the applicable scope, we can classify routing protocols into interior routing protocols and exterior routing protocols. Interior routing protocols work within a single routing domain. The classical examples are distance vector (e.g., Routing Information Protocol, or RIP) and link state (e.g., Open Shortest Path First, or OSPF). Exterior routing protocols work among separate autonomous systems, and the best example is BGP (Border Gateway Protocol).

Routing protocols that work well in wired networks may perform poorly in MANETs, if we directly use them without significant changes. The following differences between MANETs and traditional wired networks make routing in MANETs very challenging:

- *Peer-to-peer operation*. MANETs work in an infrastructure-less manner. No dedicated router or central authority exists. Every node is potentially a router, not only transmitting and receiving its own packets, but also forwarding packets for other nodes.

- *Dynamic topology*. Topology is dynamic in MANETs due to node mobility and node failure. Routes may break often during a communication session, whereas in traditional networks, topology is relatively static. Traditional routing protocols may fail to converge when directly applied to MANETs.

- *Unreliable radio channel*. The following characteristics of MANETs' link layers affect the routing decision. First, interference is an important issue in MANETs. For example, communication between nodes A and B interferes with communication between nodes B and C, whereas in wired networks, these two communication sessions can happen simultaneously. Second, channel properties, such as capacity and error rate, change dynamically in MANETs. Third, an interface in MANETs typically works in half-duplex mode, which means it can only send or receive at one time. Fourth, channels in MANETs typically have much lower bandwidth than in wired networks.

- *Power constraint*. Power efficiency is an important issue in MANETs because mobile nodes generally are powered by battery, whereas energy source is not a problem in wired networks. Routing protocols for MANETs should operate efficiently and control power consumption.

- *Security vulnerabilities*. Routing in MANETs is generally prone to information and physical security threats. Compared to wired communication, wireless communication increases the possibility of eavesdropping, spoofing, denial-of-service, and other attacks. Moreover, nodes in MANETs typically have limited computing power, which may make traditional key management protocols too expensive to be applied.

Routing in MANETs has received considerable research interest over the past decade. Researchers have proposed many routing protocols for MANETs. Chapter 2 reviews a number of representative routing protocols. Despite all the progresses and research contributions, MANETs have not found many widespread applications, especially in civilian environments. This limited application of MANETs may raise some doubts about their practicability. We address this issue in two ways.

First, we can view MANETs as a prototype of future wireless networks that operate in a peer-to-peer manner. MANETs inherently have the property of polymorphism. By adding or removing certain constraints, such as mobility and power supply, we can transform a MANET into different variations: for example, a mesh network, a delay-tolerant network, or a sensor network. These three types of wireless networks have been deployed, e.g., Roofnet [2], NetEquality [3], Interplanetary Internet [4], and Smart Surrogates [63]. Sometimes these networks are even irreplaceable due to cost or deployment environment. In order to gain insights into these peer-to-peer wireless networks and improve their applicability and practicability, it is important to identify and address fundamental problems and shortcomings in MANETs. Such knowledge will potentially benefit all peer-to-peer wireless networks.

Second, the widespread use of a technology requires complex conditions, involving social, technical, and economic areas. The impact of ad-hoc networking on traditional networking is somehow analogous to the impact of peer-to-peer architecture on traditional client-server architecture. From a technical point of view, scalability, efficiency, and routing metrics are still three major problems in routing protocols for MANETs. We explain these problems in detail next.

1.2.1 Scalable Routing Protocols

Wireless networks are rapidly becoming a commonplace: laptops and PDAs have Wi-Fi cards built in as a standard configuration; more and more places, such as campuses, offices, and hotels, provide wireless Internet access. Soon, numerous mobile and embedded devices will join the revolution of pervasive computing. Existing infrastructure-based networks may become infeasible to support all of these new participants because the number of wireless devices is increasing exponentially. We envision future large wireless networks in which hosts communicate with each other in an ad-hoc manner.

The following is an example of future large MANETs. In an area such as a campus or a city, portable devices (laptops, PDAs, smart phones, etc.) communicate with each other according to a set of ad hoc networking standards, and a number of access points may be installed to form mesh networks. In such an environment, talking to anyone digitally from anywhere (within the area) at no or little cost will become a reality. There are a number of ongoing projects that aim to study and prototype large scale ad hoc networks: the ExScal project at Ohio State University [5], the Terminodes project in Switzerland [6], and the Roofnet project at MIT [2].

We can classify routing protocols for ad-hoc networks into flat and hierarchical categories based on whether routing protocols organize nodes into some form of hierarchy. Flat routing protocols assign all nodes the same functionalities, whereas hierarchical routing protocols divide a network into groups and assign a subset of nodes special functionalities, typically coordinating roles.

Flat routing protocols, such as DSR (dynamic source routing) [58] and AODV (ad-hoc on-demand distance vector) [85], work well for small networks containing tens to a few hundreds of nodes. However, their performance degrades rapidly when the network size grows. This degradation occurs because a local topology change may affect the entire network and cause excessive routing overhead. Researchers have proposed a number of approaches, such as expanding ring search [83] and query localization [24], to improve the efficiency of routing protocols for ad-hoc networks, and thus improve their scalability. However, the improvement may not be dramatic, since flat routing protocols are inherently not scalable.

Researchers design hierarchical routing protocols to solve the scalability problem. These protocols typically divide nodes in a network into subsets, e.g., clusters or zones. A topology change in one subset only affects the current subset and perhaps a few neighboring subsets. Other subsets in the network remain untouched. Thus, the hierarchy achieves the goal of scalability. Hierarchical protocols, however, have their drawbacks. They require periodic messages from each node in order to maintain the hierarchies. These periodic messages result in higher control and processing overhead, as well as increased bandwidth usage and longer delays. Moreover, leader nodes of clusters take more responsibility in routing, and thus become performance bottlenecks and may drain their batteries early.

Therefore, one objective of this dissertation is to develop a new scalable routing protocol for ad-hoc networks. This protocol should satisfy the following requirements:

- *Hierarchical structure*. Since flat routing protocols are inherently not scalable, some form of hierarchy is necessary to achieve scalability in ad-hoc networks.

- *Low overhead*. The maintenance of hierarchies requires extra control overhead. However, the overhead should be as little as possible. Particularly, the routing protocol should minimize or avoid periodic hello messages.
- *Uniform operation*. The routing protocol should not have performance bottlenecks after applying a hierarchy. In the short term, nodes involved in the high-level of a hierarchy take more responsibility. However, in the long term, nodes should perform uniformly, i.e., switch their roles in the hierarchical structure.

1.2.2 The Efficiency of Routing Protocols

In order to improve the efficiency of routing protocols for MANETs, we may tackle various aspects such as energy, memory, processing, and control overhead. This dissertation focuses on reducing the control overhead of routing protocols. Once we reduce the control overhead, we obtain improvements in other aspects as well.

We can classify routing protocols for MANETs into proactive and reactive categories based on the manner in which the protocols establish and maintain routes. Proactive routing protocols require nodes to exchange routing information (e.g., one-hop neighbors of a node) periodically and compute routes continuously between any two nodes in the network, regardless of whether the routes will be used or not. This scheme wastes resources such as energy and bandwidth in MANETs, which are resource-constrained. On the other hand, reactive (or on-demand) routing protocols don't exchange routing information periodically. Instead, they discover a route only when it is needed by the communication between two nodes. In these protocols, a source node discovers a route to a destination node typically by flooding the entire network or a part of the network with a route request (RREQ) message. The destination sends a route reply (RREP) message to the source after receiving the RREQ. Previous work [23, 57] shows that on-demand routing protocols perform better than proactive routing protocols. Research in the literature mainly focuses on on-demand routing protocols.

Although existing on-demand routing protocols such as DSR and AODV work well for small MANETs, they still cause significant control overhead to the network. The dominating control overhead comes from the route-discovery process. One route discovery may incur tens, even hundreds of transmissions for the RREQ message. Moreover, since the network topology is dynamic, routes may break often, and each break typically triggers a new route-discovery process. Therefore, an approach that considerably reduces the overhead of route discovery is likely to improve the efficiency of on-demand routing.

This dissertation identifies three problems in existing on-demand routing protocols for MANETs. These findings are both original and important for routing in wireless ad-hoc networks. The following subsections discuss each of these findings: loss of route reply messages, in-transit route request packets, and the unilateral operation of traditional route discovery schemes.

Loss of Route Reply Messages

An on-demand routing protocol typically consists of two components: route discovery and route maintenance. Route discovery happens when a source node (say S) has data packets to send to a destination node (say D) but S doesn't have a route to D in its routing table. To establish a route to D, S broadcasts a *route request (RREQ)* message searching for D. The RREQ message propagates throughout the entire network or to a limited scope, based on the TTL (time to live, generally using hop count) of the RREQ. When the RREQ reaches the destination D, D sends a *route reply (RREP)* message to S. Other intermediate nodes that have a route to D in their routing tables may send a *cached* RREP to S. Nodes transmit RREP using unicast. When S receives the RREP, it discovers a route to D and uses this route to send data packets to D. Route maintenance deals with managing routing information at nodes, and typically involves three operations: handling route errors, deleting stale route entries, and learning new routes from the traffic.

Due to the dynamic nature of MANETs, links between nodes tend to be ephemeral — new connections occur often but they exist only for a short time. Link failure causes packet losses, and the losses inevitably include route reply packets. Among existing on-demand routing protocols, no protocol takes special care to prevent a route reply message from being lost. When a node cannot deliver a route reply message to the intended next-hop node, the common response is to discard the message and send a route error message to the destination (initiator of the RREP). In our opinion, simply discarding undeliverable RREP messages is very wasteful, because a RREP message has a lot at stake, i.e., a RREP message represents a considerable amount of route discovery overhead. If we can salvage the undeliverable RREP message and send it to the source (possibly with a repair to the route), we can reduce route discovery overhead and achieve a noticeable performance improvement.

Utilizing In-Transit Route Request Messages

On-demand routing protocols discover a route by flooding the entire or a part of the network with RREQ packets. Routes may break due to factors such as node mobility and channel congestion. Upon each route breakage, the source node may have to conduct a new route discovery. Therefore, the dominant control overhead is RREQ packets. Usually, at any time, there are a considerable number of RREQ packets in-transit in the network, and a node relays many RREQ packets for other source nodes that are conducting a route discovery.

The above observation leads to the following intuition: When a node has to find a route to a destination, instead of immediately injecting a new RREQ message into the network, the node may utilize the RREQ packets that it relays for other nodes, because some of these packets may reach the destination to which the current node needs a route. The node tries to discover a route by attaching its request to in-transit RREQ packets that it relays. This approach has the potential to improve the performance by reducing the overhead, congestion, and power consumption at nodes.

Unilateral Route Discovery

In MANETs, routes break and need to be rediscovered often. Related work [93] shows that the average life of a path is fairly short: e.g., less than 7 seconds. Therefore, the control overhead of on-demand routing mainly comes from route discoveries, and the efficiency of the route discovery scheme directly determines the performance of routing protocols.

Traditionally, the source node takes most responsibility of discovering a route. The source determines parameters such as time-to-live (TTL) and waiting time, and broadcasts a RREQ message. The destination simply responds to a RREQ with a RREP message. For this reason, researchers also call on-demand routing protocols *source-initiated* on-demand routing protocols [92]. This dissertation calls the traditional manner of route discovery *unilateral route discovery (URD)*.

The unilateral operation is not balanced because one party bears more burden than the other. It is not efficient and the delay is long. If we can conduct route discoveries in a more balanced way, we will greatly improve the efficiency of on-demand routing. Therefore, a natural question is: “Can the source and the destination concurrently perform a route discovery?”

1.2.3 Routing Metrics

When selecting a route from a source to a destination, existing routing protocols most commonly use hop count as the path metric. However, the hop count metric may result in poor performance in networks such as wireless mesh networks (WMNs) [11], because a route with minimum hop count may include slow or lossy links, leading to poor end-to-end throughput. Moreover, nodes in WMNs typically are stationary, and routes, including low throughput ones, tend to last much longer than in mobile networks.

Researchers have proposed a number of metrics for selecting routes with high throughput in multi-hop wireless networks. De Couto et al. propose the ETX (expected transmission count) metric [31]. The ETX of a link is the expected number of transmissions to send a packet over that link, and the ETX of a path is the sum of the ETX of each link on the path. Draves et al. propose the ETT (expected transmission time) metric to find high throughput paths in heterogeneous, multi-radio WMNs [34]. We discuss more metrics in Chapters 2 and 7. This dissertation collectively calls these metrics that select routes with high throughput in multi-hop wireless networks *High-Throughput Metrics (HTMs)*.

In order to support HTMs, on-demand routing protocols need to collect the metric of each link on a path [31, 34]. Thus, protocols that use source routing are suitable to support HTMs, because they contain individual link information in their route discovery messages. As a result, DSR [58] is a natural choice when supporting an HTM. Both ETX and ETT have been implemented as a metric in DSR [31, 34]. However, the current design for supporting HTMs in DSR is not efficient, because it collects link state information and runs Dijkstra's shortest path algorithm.

AODV [85] is another well-known and well-studied on-demand routing protocol. However, when trying to support an HTM in AODV, we face problems because control messages in AODV typically do not contain individual link information.

This dissertation aims to design an efficient and generalized scheme to support HTMs in on-demand routing protocols, including AODV.

1.3 Contributions of the Dissertation

In response to the problems discussed in the previous section, this dissertation presents a number of new protocols to solve the scalability and the efficiency problems in designing routing protocols for MANETs. Contributions of the dissertation include a scalable routing protocol, three routing optimization techniques, and a scheme to support high-throughput routing metrics.

1.3.1 Way Point Routing Model (WPR) and DSR Over AODV Routing Protocol (DOA)

We have developed a lightweight hierarchical routing model called Way Point Routing (WPR) for MANETs [15]. WPR selects a number of nodes on a route as waypoints and divides the route into segments at the waypoints. Waypoints run a high-level inter-segment routing protocol. Nodes on each segment run a low-level intra-segment routing protocol. When an intermediate node moves away or fails, previous protocols discard the whole original route and discover a new route, but WPR only requires the two waypoints of the broken segment to repair that segment.

In addition, we have developed an instantiation of WPR, in which we use DSR as the inter-segment routing protocol and AODV as the intra-segment routing protocol. We call this instantiation the DSR Over AODV (DOA) routing protocol [15]. DOA combines DSR and AODV, two well-known on-demand routing protocols, in a hierarchical manner and includes each of these protocols as special cases when the segment size is minimal or maximal. Furthermore, we have designed two novel techniques that apply to DOA: an efficient loop-detection method and an inter-segment route repair optimization.

1.3.2 Salvaging Route Reply (SRR) for On-Demand Routing Protocols

We have developed the Salvaging Route Reply (SRR) algorithm to prevent the loss of route reply messages. SRR has two schemes: SRR1 [14] and SRR2 [19]. In SRR1, a node actively broadcasts a salvage request message to its one-hop neighbors, asking for a cached path to the source. SRR2 is an improvement over SRR1. In SRR2, intermediate nodes passively maintain a backup path to the source utilizing duplicate RREQ packets. When the RREP cannot be relayed using the original path to the source, the node directly switches to the backup path. Therefore, SRR introduces very little extra overhead (in scheme one) or no extra overhead (in scheme two) to the routing protocol, but it has potential to substantially reduce the overhead of route discovery and significantly improve other performance metrics such as packet delivery ratio and end-to-end delay.

1.3.3 Carpooling in MANETs: the Case of Multiple-Target Route Discovery

Carpooling is an efficient transportation option in real life because it reduces traffic and cost. We have applied this idea to MANETs and have developed Multiple-Target Route Discovery (MTRD) [17]. MTRD aggregates multiple route requests into one RREQ message and discovers multiple targets simultaneously. When a node has to find a route to a destination, instead of immediately injecting a new RREQ message into the network, the node tries to discover a route by attaching its request to in-transit RREQ packets that it relays for other nodes. MTRD improves routing performance by reducing the number of regular route discoveries.

1.3.4 Route Discovery in Mobile Ad Hoc Networks: from Unilaterality to Bilaterality

The unilateral operation of route discovery is unbalanced and is not ideal for MANETs. We have designed a new scheme called Bilateral Route Discovery (BRD), in which both source and destination actively participate in the process of discovering a route between them [16, 18]. BRD has the potential to halve the control overhead. As an underlying protocol for BRD, we developed Gratuitous Route Error Reporting (GRER). GRER bypasses a failed link and notifies the destination of a broken route. The destination can thus play an active role in the ensuing route re-discovery.

1.3.5 Supporting High-Throughput Routing Metrics in Multi-Hop Wireless Networks

We have designed an efficient and generalized approach called Accumulated Path Metric (APM) for supporting high-throughput metrics (HTMs) in on-demand routing protocols [20]. One advantage of APM is that it is able to find the shortest path, in terms of a particular metric, without collecting topology information and without running a shortest-path algorithm. APM significantly simplifies supporting HTMs in DSR. We have proved that with reasonable assumptions, APM discovers the shortest path. In addition, we have addressed the problem of duplicate RREQ transmissions with existing HTM schemes and have developed a Broadcast Ordering (BO) technique to suppress unnecessary RREQ transmissions. Furthermore, we have given a taxonomy of existing schemes that support high throughput metrics in wireless ad-hoc networks.

1.4 Organization of Dissertation

The remainder of this dissertation is organized as follows. The next chapter presents background information and reviews related work. We first review routing protocols that are suitable for small MANETs. We then review routing protocols that scale to large MANETs. Moreover, we discuss a wide range of approaches to improving routing efficiency. Finally, we present different metrics for measuring the quality of a link and a path.

Chapters 3–7 form the technical core of this dissertation. Chapter 3 presents the WPR routing model and the DOA routing protocol. Chapter 3 also presents two techniques for DOA: a loop detection method and an inter-segment route repair optimization.

Chapter 4 presents the idea of salvaging route reply messages and presents two schemes of SRR. SRR1 uses a one-hop route discovery to find an alternative path. SRR2 is an improvement over SRR1 that does not require extra control messages. SRR2 uses duplicate RREQ packets to save a backup path to the source node.

In Chapter 5, we observe that carpooling saves energy and reduces congestion in real life. We exploit this idea in MANETs to improve the efficiency of routing protocols. As a beginning work towards this direction, we present MTRD (multi-target route discovery).

In Chapter 6, we observe that traditional unilateral route discovery is not balanced and not ideal for MANETs. We propose BRD (bilateral route discovery) in which both source and destination nodes actively participate in a route discovery process. To notify a destination of a link failure, we propose GRER (gratuitous route error reporting).

Chapter 7 presents a generalized and efficient scheme called APM (accumulated path metric) to support high-throughput metrics in on-demand routing protocols for MANETs. We prove the correctness of APM and also present the BO (broadcast ordering) technique to unnecessary RREQ transmissions.

Finally, Chapter 8 concludes this dissertation and gives future research directions.

Chapter 2

Related Work

Researchers have proposed many routing protocols for MANETs. We can classify these protocols into different categories according to different criteria. If based on communication patterns (one-to-one, one-to-many, one-to-all), we can classify routing protocols for MANETs into unicast, multicast, and broadcast protocols. Most research work in the literature focuses on unicast routing, which is the primary interest of this dissertation as well.

Based on how routing protocols establish and maintain routes, we can group them into proactive (or table-driven) protocols and reactive (or on-demand) protocols (though a few hybrids exist). Proactive protocols propagate topology information periodically and find routes continuously between any two nodes in the network, whereas reactive protocols find routes only when they need routes to send data packets. Performance analysis and simulation results [23, 57] show that reactive protocols outperform proactive protocols in terms of packet delivery ratio, routing overhead, and energy efficiency.

Based on whether requiring nodes to have locationing capability, e.g., using GPS (Global Positioning System) devices, we can classify routing protocols into position-based and topology-based protocols. Given the position of a destination node, position-based routing protocols can simply use greedy forwarding, i.e., sending packets to nodes that are closer to destinations. However, providing an efficient and scalable location service in MANETs is difficult.

Based on whether using a hierarchical structure on nodes in the network, we can classify routing protocols into flat and hierarchical protocols. In flat routing protocols, all nodes have the same role and use the same algorithm. Flat routing protocols are suitable for small networks. In hierarchical routing protocols, a subset of nodes assumes more routing responsibility than other nodes. Hierarchical routing protocols are suitable for medium to large networks.

Table 2.1 lists a number of routing protocols for ad-hoc networks and their classifications. The rest of this chapter is organized as follows. In the next section, we review routing protocols that are suitable for small MANETs (Section 2.1). In section 2.2, we review routing protocols that can scale to large MANETs. Section 2.3 discusses a wide range of approaches to improving routing efficiency. Section 2.4 presents different metrics for measuring the quality of links and paths.

2.1 Routing Protocols for Small Networks

This section reviews the following routing protocols: dynamic source routing (DSR), ad-hoc on-demand distance vector (AODV), temporally-ordered routing algorithms (TORA), destination sequenced distance vector (DSDV), fisheye state routing (FSR), and location-aided routing (LAR). We select these protocols for the following reasons: DSR and AODV are two of the best known on-demand routing protocols; TORA presents a novel idea; DSDV and FSR are two representative proactive routing protocols; and LAR is a representative position-based routing protocol.

2.1.1 Dynamic Source Routing (DSR)

DSR [58] uses the source routing approach, in which source nodes include the complete route (an ordered list of nodes, along which packets can reach their destinations) in the header of data packets. DSR uses three types of control messages: *route request (RREQ)*, *route reply (RREP)*, and *route error (RERR)*. When a source needs to find a route to a destination, it floods the network with a RREQ message. The RREQ records the path it has traversed. Upon receiving a RREQ message, intermediate nodes (neither the source nor the destination) typically append their addresses in the RREQ and relay it using broadcast. Intermediate nodes may send a RREP to the source if they know a route to the destination. When the route request reaches the destination, the destination initiates a RREP containing the path recorded by the RREQ and sends the RREP to the source using unicast. The RREP is relayed to the source by intermediate nodes through the reverse path discovered by the RREQ, or through another route from the destination's route cache. When the source receives such a reply, it can use the discovered route to send data packets. When a link on a route breaks, the upstream node detects that its downstream node is not reachable and sends a RERR message to the source. The source subsequently may initiate another route discovery to establish a new route to the destination.

2.1.2 Ad-hoc On-demand Distance Vector (AODV) Routing Protocol

AODV [85] is based on the traditional distance vector routing mechanism and discovers routes reactively. AODV also uses messages RREQ, RREP, and RERR. When a source node needs to

Table 2.1: Classification of routing protocols for ad-hoc networks.

Protocol	*Reactive (RE), Proactive (PR), Hybrid (HY)*	*Position-based (PB), Topology-based (TB)*	*Flat (FL), Hierarchical (HR)*	*Link-State (LS), Distance Vector (DV), Source Routing (SR)*
ABR (Associativity-Based Routing) [99]	RE	TB	FL	SR
AODV (Ad-hoc On-Demand Distance Vector) [85]	RE	TB	FL	DV
CEDAR (Core Extraction Distributed Ad-hoc Routing) [95]	RE	TB	HR	LS
CGSR (Cluster-Head Gateway Switch Routing) [28]	PR	TB	HR	DV
DSDV (Destination-Sequence Distance Vector) [84]	PR	TB	FL	DV
DSR (Dynamic Source Routing) [58]	RE	TB	FL	SR
FSR (Fisheye State Routing) [79]	PR	TB	FL	LS
GPSR (Greedy Perimeter Stateless Routing) [61]	RE	PB	FL	—
GSR (Global State Routing) [26]	PR	TB	FL	LS
HSR (Hierarchical State Routing) [81]	PR	TB	HR	DV
LANMAR (Landmark Ad-hoc Routing Protocol) [80]	PR	TB	HR	LS, DV
LAR (Location-Aided Routing) [64]	RE	PB	FL	SR
OLSR (Optimized Link State Routing) [56]	PR	TB	FL	LS
SSA (Signal Stability-based Adaptive Routing) [35]	RE	TB	FL	SR
STAR (Source-Tree Adaptive Routing) [41]	PR	TB	FL	LS
TORA (Temporally-Ordered Routing Algorithm) [77]	RE	TB	FL	—
WRP (Wireless Routing Protocol) [73]	PR	TB	FL	DV
ZRP (Zone Routing Protocol) [49]	HR	TB	HR	DV, SR

find a route to a destination, the source broadcasts a RREQ message. When intermediate nodes propagate the RREQ in the network, these nodes learn routes to the source because AODV assumes symmetric links. Upon receiving the RREQ, the destination sends a RREP message to the source. Intermediate nodes that have a *fresh* route to the destination may also send a “cached” RREP to the source. AODV uses the *sequence number* of the destination to determine the freshness of a route. Intermediate nodes relay the RREP to the source using unicast along the path learned when they propagated the RREQ. Thus, intermediate nodes establish the route from the source to the destination when they forward the RREQ and establish the route from the destination to the source when they relay the RREP. AODV maintains a route in a distributed manner, i.e., each node on the route (except the destination) keeps a pointer to its next hop. When a route breaks at a link, the upstream node detects the link break because the downstream node is not reachable. The upstream node sends a RERR message to the source, and the source subsequently may initiate another route discovery.

2.1.3 Temporally-Ordered Routing Algorithms (TORA)

TORA [77] is based on the concept of link reversal [40]. An important design goal of TORA is to minimize the reaction to topology changes. A topology change (e.g., a link break) typically affects a small set of nodes near the change. TORA does not consider route optimality (e.g., the shortest path) of primary importance. It may use longer routes to avoid discovering newer routes.

TORA defines the routes to a destination by a directed acyclic graph (DAG) rooted at the destination. In the DAG, TORA assigns each node a height value with respect to the destination. Time is also a part of the height value, and thus TORA assumes synchronized clocks among all nodes. TORA assumes bi-directional links, but in order to form the DAG, TORA assigns each link a logical direction from the node with a higher height to the node with a lower height.

We can describe the routing in TORA in the form of water flowing downhill from the source (with higher height value) towards the destination (with lower height value). TORA creates routes using *QUERY* and *UPDATE* messages. When a source needs to find a route to a destination, it broadcasts a $QUERY$ message, and neighbor nodes propagate the message outward. On receiving the $QUERY$, a node that has a route to the destination or the destination itself broadcasts an $UPDATE$ message containing its own height. On receiving the $UPDATE$, each node that doesn't have a route to the destination updates its height to add an out-going edge to the DAG with respect to the destination. TORA maintains routes through height adjustment and $UPDATE$ exchange. If a node cannot forward a packet to its next hop because of link failure, TORA sets the height of the node to the local highest. Then the packet will flow back out of the node. TORA uses a $CLEAR$ message to erase invalid routes after detecting a network partition.

2.1.4 Destination Sequenced Distance Vector (DSDV) Routing Protocol

The DSDV [84] routing protocol is based on the classical Bellman-Ford routing algorithm. A major improvement of DSDV is that it ensures loop freedom by using the sequence number. Each node maintains a routing table ($\langle destination : next\_hop \rangle$), in which all the possible destinations in the network are recorded as entries. The table also records the number of hops to each destination and a sequence number for each destination. DSDV uses the sequence number to compare the freshness of routes.

In DSDV, nodes periodically transmit routing-table updates throughout the network in order to maintain the consistency of routing tables. To reduce the update overhead, DSDV defines two types of updates: periodic full update and event-driven incremental update. On receiving a route update, a node updates its routing table according two rules: first, always use the route with a fresher sequence number; second, use the route with same sequence number but better metric (e.g., lower hop count).

To prevent fluctuations of routing table advertisements (e.g., due to unstable routes), DSDV introduces *settling time*. A node advertises a new route to its neighbors only if the settling time of the route has expired and the route remains stable.

2.1.5 Fisheye State Routing (FSR)

FSR [79] is based on the observation that topology changes in a region have more effect on routing locally in the region, and they have less effect on routing in regions far away. Therefore, a routing protocol does not need to frequently update topology changes in one region to nodes in remote regions.

With respect to a particular node, FSR divides the network into different scopes, i.e., inner scope and outer scope, according to the number of hops from this node to other nodes. A node broadcasts its link-state information to its neighbors, but the node advertises route entries for nodes belonging to different scopes at different frequencies. The node advertises routing entries for its inner scope at the highest frequency and advertises other entries at lower frequencies. Therefore, nodes will have more accurate link-state information for local nodes and less accurate information for remote nodes. To further reduce routing overhead, nodes only periodically broadcast their link-state information. Events such as link break do not trigger link state updates.

When a source sends a packet to a remote destination, it uses the link-state information it has collected to compute the shortest path. The path may not be accurate, especially for the part near the destination. When intermediate nodes route the packet toward the destination, they have more accurate link-state information for the destination and route the packet more appropriately. When the packet reaches the inner scope of the destination, the link-state information is up-to-date and inner-scope nodes deliver the packet correctly.

2.1.6 Location-Aided Routing (LAR)

LAR [64] assumes that nodes know their own location and the location of destinations. LAR uses the location information to limit the flooding of RREQ messages to a *request zone* that contains a small group of nodes. This approach reduces the routing overhead compared to other routing protocols such as DSR or AODV, which typically flood RREQ messages throughout the network.

Before calculating the request zone, LAR determines the *expected zone* of the destination. Suppose a source S knows a destination D was at location L at time t_0, and let the current time be t_1; the expected zone of D is the region that S expects to contain D at time t_1. For example, if S knows that D travels with average speed v, then S may estimate that the expected zone is the circular region of radius $v * (t_1 - t_0)$, centered at location L.

LAR uses the request zone to limit the propagation of RREQ messages. A node forwards a RREQ only if it is located in the request zone. The request zone includes the expected zone described above and other regions around the expected zone. LAR proposes two schemes to define the request zone.

In scheme 1, the request zone is the smallest rectangle that includes the current location of the source and the expected zone of the destination.

In scheme 2, nodes forward a RREQ message to neighbors that are closer to the destination's location. The source S includes two pieces of information for the destination D in the RREQ message: the coordinates (X_d, Y_d) of D that S knows at time t_0; the distance $DIST_s$ between S

and D at time t_0. When a node I receives the RREQ from S, it calculates its distance from D (using coordinates (X_d, Y_d)). We denote the distance by $DIST_i$. If $DIST_i$ is not larger than $DIST_s$, node I replaces $DIST_s$ with $DIST_i$ in the RREQ and forwards the RREQ.

When LAR fails to find a route to the destination due to estimation error or other reasons, it resorts to normal flooding.

2.2 Scalable Routing Protocols

Protocols introduced in the previous section work well for small networks, which typically have fewer than one hundred nodes. In larger networks, performance of these protocols degrades rapidly due to the routing overhead. As in traditional wired networks, researchers introduce hierarchies to MANETs to solve the scalability problem. This section reviews three scalable routing protocols, namely, clusterhead-gateway switch routing (CGSR), landmark ad-hoc routing (LANMAR), and the zone routing protocol (ZRP).

2.2.1 Clusterhead-Gateway Switch Routing (CGSR)

CGSR [28] is based on DSDV [84] and works proactively. CGSR organizes a network into *clusters* and elects a *clusterhead* in each cluster by running an efficient clustering algorithm. Other nodes in each cluster are one hop away from the clusterhead. Nodes that belong to more than one cluster are *gateways*.

CGSR uses the Least Clusterhead Change (LCC) algorithm for the clustering operation. In LCC, a clusterhead change occurs only when two clusterheads come into one cluster or one node moves out of the range of all the clusterheads.

Each node maintains a *cluster member table* and a *distance vector table*. The cluster member table records which cluster each node belongs to and stores the clusterhead of each cluster. The distance-vector table records next hops (usually gateways) to all clusters. CGSR reduces the size of the routing table by maintaining a route entry for each cluster instead of for each node. Similar to DSDV, CGSR uses the sequence number to avoid stale routes. CGSR broadcasts the cluster-member table and the distance-vector table periodically among clusterheads and gateways.

In CGSR, routing generally works in the following manner. The source node of a data packet first transmits the packet to its clusterhead. The clusterhead consults its cluster-member table to find out which cluster the destination belongs to. Then the clusterhead consults its distance-vector table to get the next hop, which is typically a gateway node. The clusterhead sends the packet to the gateway and the gateway sends the packet to the clusterhead of the next cluster. This process continues until the packet reaches the clusterhead of the destination cluster. The destination clusterhead delivers the packet to the destination. Therefore, a route in CGSR is typically of the form $Clusterhead - Gateway - Clusterhead - Gateway....$

2.2.2 Landmark Ad-hoc Routing Protocol (LANMAR)

Pei et al. [80] propose LANMAR for MANETs that exhibit group mobility. LANMAR combines the features of Fisheye State Routing (FSR) [79] and Landmark routing that was first introduced by Tsuchiya for fixed wide area networks [101]. In LANMAR, each node has a unique physical id (e.g., MAC address) and a unique logical id defined as $\langle subnet : host \rangle$. A subnet consists of nodes with common interests and group mobility. A node's host id is unique in its subnet and may use the node's physical id. LANMAR dynamically elects a landmark node as the representative of each subnet. Each node maintains two routing tables: the *local routing table* for nodes within the subnet and the *landmark table* that maintains distance vectors for all landmarks. Routing in LANMAR is a modified version of FSR. The main difference is that the FSR routing table includes all nodes in the network, whereas LANMAR routing tables only include nodes within a node's fisheye scope (local routing table) and landmark nodes in the network (landmark table). This scheme reduces routing-table size and update-traffic overhead.

In LANMAR, the header of a data packet carries the logical id of the destination. When a node relays a data packet, if the destination is within its neighbor scope, the packet can be forwarded directly. Otherwise, the node searches the logical subnet field of the destination and routes the packet towards the landmark for that logical subnet. The packet does not have to pass through the landmark of the destination's subnet (unlike CGSR). Instead, once the packet reaches the scope of the destination, it is routed to the destination directly.

The route update exchange in LANMAR is similar to FSR. Each node periodically exchanges topology information with its direct neighbors. In each update, a node sends entries within its fisheye scope. The node also piggybacks a distance vector whose size is equal to the number of logical subnets, which is also the number of landmark nodes. LANMAR uses the sequence-number mechanism to indicate the age of routing-table entries.

2.2.3 Zone Routing Protocol (ZRP)

ZRP [49] is a hybrid routing protocol that combines features of both proactive and reactive routing protocols. In ZRP, each node dynamically maintains a *zone* centered at itself. A zone is a collection of neighbors and links within a predefined number of hops called the *zone radius*. The construction of a zone requires a node to discover its neighbors. ZRP uses a separate Neighbor Discovery Protocol (NDP) for this purpose. In NDP, nodes typically broadcast periodic hello messages.

After a node establishes its zone, the node uses Intrazone Routing Protocol (IARP) to communicate with the interior nodes of its zone. IARP proactively maintains routes to destinations within the zone. To communicate with destinations outside the zone, a node uses IntErzone Routing Protocol (IERP). IERP works reactively and uses RREQ/RREP messages like other on-demand routing protocols. During the process of a route discovery, ZRP uses Bordercast Resolution Protocol (BRP) [49] to reduce the overhead of RREQ messages. BRP forwards a RREQ recursively to border nodes, which are r hops away from the current node, where r is the zone radius. This process continues until one node that has the destination in its zone is reached.

2.2.4 Dynamic Addressing

Eriksson et al. [37] propose a dynamic addressing scheme to solve the scalability problem in MANETs. In their work, the traditional IP address is used only for identification purposes, not for routing. Dynamic addressing introduces routing addresses to reflect the node's location in the network topology. Dynamic addressing organizes routing addresses into a tree. Routing addresses may change dynamically due to node mobility. A range of routing addresses with a common prefix forms a subtree, with the fundamental constraint that the nodes of a subtree form a connected subgraph in the network topology. Scalability relies on balancing the routing-address tree. In a mobile environment, the tree may get unbalanced, and the operation to optimize/balance the tree may be complicated and expensive.

2.3 Approaches to Improving Routing Efficiency

Researchers have proposed many techniques to improve the performance of routing protocols for MANETs. Some techniques may improve many protocols (e.g., expanding ring search [83]), whereas others are suitable for a particular category of protocols (e.g., route cache strategy for DSR [60]). Different techniques may address different aspects of routing in MANETs, such as reducing the overhead of route request messages (e.g., query localization [24]), or making an active route last longer (e.g., local repair [83]).

2.3.1 Expanding Ring Search

ERS (expanding ring search) [83] avoids flooding the entire network when a routing protocol conducts a route discovery. Thus, ERS reduces the overhead of RREQ messages. Under ERS, a source node broadcasts a RREQ message within successively larger areas, centered at the source, until the source receives a route reply message. Initially, the source uses a small TTL value for a RREQ: for example, 2 hops. If the source has not received a route reply by the end of the discovery time, it broadcasts another RREQ with an incremented TTL value. This process continues until the TTL reaches a threshold value. At this point, the source floods the entire network to find a route.

Sucec and Marsic [97] propose a query scope agent (QSA), which helps select an appropriate ERS that satisfies a delay requirement. Chang and Liu [25] present a dynamic programming formulation to minimize the expected cost of TTL-based flooding search.

2.3.2 Query Localization

Query localization [24] also aims to reduce the overhead of RREQ messages. This approach limits the propagation of a RREQ message to the area around the previously known route to the destination node. Therefore, routing protocols will not always flood a RREQ message through the entire network. One method to perform query localization is *exploiting node locality*. This method assumes that the destination has not moved too far from its previous known location, and hence the route-discovery process can find the destination within a few hops from the most recently used route to it. A RREQ packet contains a counter to control how far intermediate nodes can propagate it from the previous route. When a node receives the RREQ, if the node was on the previous route, it resets the counter to zero and relays the RREQ. Otherwise, the node increments the counter and relays the RREQ if the counter is no greater than a threshold value.

2.3.3 Route Repair

Route repair tries to fix a broken active route, and thus extend the lifetime of a route and reduce the number of global route discoveries. Traditionally, when a link on an active route breaks, the upstream node of the failed link sends a RERR (route error) message to the source. Upon receiving the message, the source may start another route discovery. In the local repair approach [83], instead of sending a RERR message immediately, the upstream node first attempts to repair the broken route locally. It does so by broadcasting a RREQ (route request) with a limited TTL to discover a path to the destination. When the repairing node receives a RREP (route reply) message, it means the node repairs the route successfully. Otherwise, it sends a RERR message to the source.

Goff et al. [45] propose a preemptive routing scheme. When a route is likely to break, the source may initiate a route discovery early to avoid a disconnection. This approach reduces end-to-end delay with a small increase in control overhead. Costa et al. [30] propose a controlled flooding (CF) approach to establish alternative paths around an original route. When the original route breaks, a secondary path is used to forward data packets.

2.3.4 Path Optimization

Path optimization [44, 47] is based on the following observation: routes are optimal (e.g., in terms of number of hops) during the establishment phase, but they may become sub-optimal over time due to node mobility. Path optimization approaches typically require nodes to work in promiscuous mode, in which a node processes all packets it receives rather than just packets addressed to it, to find an optimization opportunity. In SHORT [47], data packets carry two additional fields: *hop_count* (to the destination) and *sending_node* (the node that transmits the packet). Each node maintains a *comparison array* to collect information from data packets that the node receives or overhears. When a node receives or overhears a data packet, it compares the packet to the information in its comparison array. If an optimization is possible, the node sends a message to notify the node from which it receives the packet. PCA [44] improves SHORT by adding another field to the header of data packets: hop count to the source. Thus, when a node receives or overhears a data packet, it attempts to optimize the route in both directions, i.e., to the destination and to the source.

2.3.5 Route Caches

In order to reduce the number of route discoveries, routing protocols may cache routes that they previously discovered or overheard from the traffic. However, such caching requires proper strategies for managing the structure and contents of route caches. Hu and Johnson [52] study the strategies for cache structure, cache capacity and cache timeout. They propose a link cache structure and several link timeout algorithms. Lou and Fang [67] propose an adaptive link timeout mechanism, which adjusts the link lifetime according to the real link lifetime statistics. Marina and Das [69] propose three techniques to improve cache correctness in DSR: wider error notification, route expiry mechanism with adaptive timeout selection, and negative caches. Yu and Kekem [107] propose a cache update algorithm to make route caches adapt to topology changes without using ad-hoc parameters such as mobility models.

2.3.6 Path Stability

Path stability is an important goal for MANETs, because a stable path lasts longer and thus reduces the number of route discoveries. Several routing protocols select paths based on path stability: associativity based routing (ABR) [99], signal stability adaptive routing (SSA) [35], route lifetime assessment based routing (RABR) [10], and flow oriented routing protocol (FORP) [96]. The estimation of path stability either uses past topology changes (e.g., ABR and SSA), or predicts future topology changes (e.g., RABR and FORP). Strategies that predict future topology changes perform better [72]. However, these strategies require nodes to know geographic information such as location, and velocity.

Han et al. [50] study the distribution of path duration. First, they prove that, under a number of mild conditions, when the hop count of a path is large, the distribution of path duration can be approximated by an exponential distribution. Then, they prove that the parameter of the exponential distribution is related to the link durations only through their means, and they calculate the parameter by the sum of the inverses of the expected link durations. Finally, they propose a scheme for existing routing protocols to select the paths with the largest expected durations.

2.3.7 Identifying Mobility-Based Link Failure

Routing protocols use failure notification from the MAC layer or use a periodic HELLO protocol to detect failed links. However, existing routing protocols or MAC layer protocols do not distinguish between a link failure caused by mobility and a link failure caused by congestion. If a link failure is caused by congestion, we may let the transport protocol reduce its rate, instead of letting the routing protocol run an expensive route discovery. Pandey et al. [75] propose the mobility detection algorithm (MDA), which uses MAC-layer statistics to distinguish between mobility and congestion-based failures. With MDA, routing protocols only react to link failures due to mobility.

2.4 Routing Metrics

Routing protocols such as AODV [85], DSR [58], DSDV [84], and TORA [77] have traditionally used hop count as the metric when selecting routes. The advantage of the hop-count metric is its simplicity, whereas the disadvantage of hop count metric is that it does not consider link quality. Using routes that contain slow links may lead to poor performance in wireless networks. A number of researchers have recognized this issue [9, 12, 31, 35, 46, 53, 104, 105]. This section first reviews routing approaches that take into account signal strength, and then reviews existing high-throughput routing metrics. Chapter 7 presents a detailed taxonomy of existing high-throughput routing metrics.

2.4.1 Approaches Based on Signal Strength

In wireless networks, signal strength offers the most direct estimate of the ability of nodes to communicate with each other. Thus, routing protocols often use signal strength as an indication of link quality. Dube et al. [35] propose SSA (signal stability adaptive routing), where a node classifies its neighbors as strongly or weakly connected according to the criteria of signal strength and location stability. Generally, nodes drop route requests from weakly connected neighbors. Lundgren et al. [68] introduce *communication gray zones* to refer to the zones in which nodes can hear HELLO packets but cannot exchange data packets. They propose three work-arounds to filter out gray zone links. Zhao and Govindan [108] observe that assessing the packet loss of a link from signal strength alone may not work well. High signal strength usually implies low packet loss, but low signal strength does not imply high packet loss. Compared to the schemes in the next subsection, these approaches seem to be coarser in measuring the quality of a link, because they treat a link as either good or bad. As a result, they may eliminate links that are necessary for connectivity.

2.4.2 High-Throughput Routing Metrics

Researchers have proposed a number of metrics to measure the quality of a wireless link. A representative of such metrics is ETX (expected transmission count) [31]. The ETX metric represents the expected number of transmissions needed to send a unicast packet over a link. The calculation of ETX of a link is based on the measurements of the delivery ratio in both forward and reverse directions, denoted as d_f and d_r, respectively. Then, the calculation of ETX metric uses the formula: $ETX_{link} = 1/(d_f \times d_r)$.

Draves et al. [34] propose the ETT (estimated transmission time) metric for networks where nodes are equipped with multiple radios and communicate on multiple channels. A path metric called WCETT (weighted cumulative ETT) combines the ETTs of individual links on a path. WCETT takes into account both the sum of ETTs and the channel diversity.

Adya et al. [9] propose RTT (round trip time) metric. In RTT, a node sends a unicast probe packet carrying a timestamp to each of its neighbors every 500 milliseconds. Each neighbor immediately responds to the probe with an acknowledgment echoing the timestamp. Thus, the sending node can measure the RTTs to each neighbor.

Draves et al. [33] implement and compare four routing metrics, namely, ETX [31], RTT [9], per-hop packet pair delay (PktPair) and hop count. They study these metrics in the link quality source routing (LQSR) protocol, which is based on DSR. The results show that ETX performs best in stationary networks. One observation is that hop count metric performed well in a mobile scenario, because it reacts quickly to topology changes.

Lee et al. [66] propose the NADV (normalized advance) metric for geographic routing in multi-hop wireless networks. NADV denotes the amount of advance achieved per unit cost. Instead of using distance progress as a metric, NADV selects the next hop with the trade-off between proximity and link cost. In NADV, a routing protocol can use various approaches, such as packet error rate, link delay, and energy consumption, to define link cost.

Chapter 3

DOA: DSR Over AODV Routing

3.1 Introduction

Based on whether nodes in the network are organized in a hierarchical structure, we can classify routing protocols into flat and hierarchical protocols, as discussed in Chapter 2. When a route breaks due to node mobility or node failure, flat routing protocols like DSR and AODV typically discard the whole original route and initiate another round of route discovery to establish a new route from the source to the destination. However, when a route breaks, usually only a few hops are broken, but other hops are still intact. Discarding the whole route wastes the knowledge of the original route and causes considerable overhead in global route discoveries.

An optimization to AODV is local repair, as described in an Internet draft [83]. However, local repair is only suitable for situations where link failures occur near the destination. In Section 6.12 (Local Repair) of AODV draft [83], "when a link break in an active route occurs, the node upstream of that break MAY choose to repair the link locally if the destination was no farther than MAX_REPAIR_TTL hops away". The reason for this limitation is that intermediate nodes only know the destination and the next hop for a route, and the target of the local repair has to be the destination. If a link failure occurs far from the destination, it would be better for the source to discover a new route directly.

Therefore, the main motivation of this work is as follows: By establishing and maintaining active routes hierarchically, a routing protocol can repair a broken route locally wherever the link failure happens along the route and reduce the number of global route discoveries. Consequently, the routing protocol reduces routing overhead and improves scalability and performance.

In this chapter, we present a scalable routing model for MANETs, Way Point Routing (WPR), which maintains a hierarchy only for active routes. WPR selects a number of intermediate nodes on a route as way points and divides the route into segments at the way points. WPR also considers the source and the destination as way points. Way points run a high-level inter-segment routing protocol, and nodes on each segment run a low-level intra-segment routing protocol. One distinct advantage of WPR is that when a node on the route moves out or fails, instead of discarding the whole original route and discovering a new route from the source to the destination, only the broken segment has to find a new path. This approach has a clear performance advantage, such as low routing overhead and low end-to-end delay.

Compared to hierarchical routing protocols such as CGSR [28] and Zone Routing Protocol (ZRP) [49], WPR is a light-weight scheme for two reasons. First, the hierarchy in WPR only involves nodes on active routes, whereas the hierarchy in CGSR and ZRP involves all nodes in the network. Second, the hierarchy in WPR is easier to maintain. The hierarchy in CGSR and ZRP is more complicated because it is built in two dimensions, i.e., the network is divided into regions as shown in Figure 3.1, whereas the hierarchy in WPR is built in one dimension, i.e., a route is linearly divided into segments as shown in Figure 3.2. Moreover, nodes in WPR work uniformly because the assignment of waypoint nodes and non-waypoint nodes is specifically for each route, and a node may act as a waypoint node for one route and act as a non-waypoint node for another route. This feature facilitates the uniform consumption of resources at nodes in MANETs. In hierarchical routing protocols such as CGSR, nodes work non-uniformly, and a center node may become a traffic bottleneck and drain its battery early.

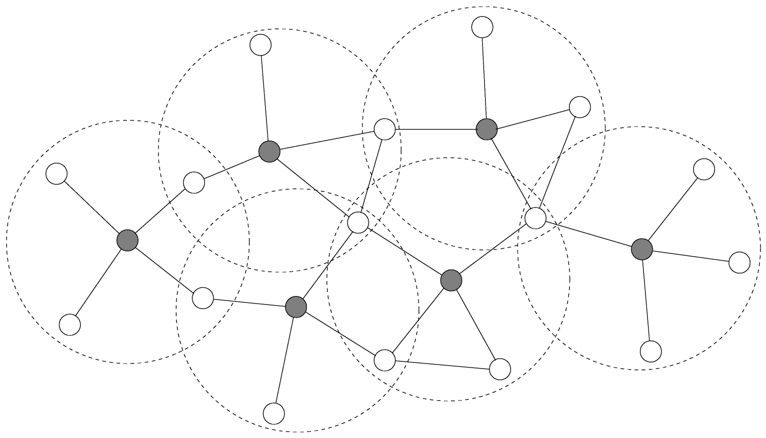


Figure 3.1: A hierarchy in two dimensions.

As an instantiation of WPR, we choose DSR and AODV as the inter-segment and the intra-segment protocols, respectively. We term this instantiation the DSR Over AODV (DOA) routing protocol. Thus, DOA combines the two standard routing protocols for MANETs, DSR and AODV, hierarchically. Moreover, they become two special cases of DOA. When the segment length is one, DOA works in DSR mode because each hop is a segment and the inter-segment routing protocol (DSR) dominates. When the segment length is a large number, DOA works in AODV mode because the whole route is one segment and the intra-segment routing protocol (AODV) dominates. In this chapter, we also present two novel techniques for DOA, one is an efficient loop detection method and the other is a multi-target route discovery.

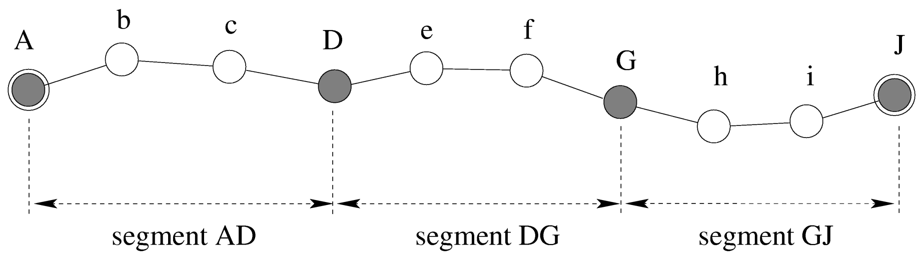


Figure 3.2: WPR divides a route into segments.

The remainder of this chapter is organized as follows. In the next section, we present the Way Point Routing model. In Section 3.3, we describe the DOA routing protocol. Section 3.4 presents simulation setup and results, and Section 3.5 summarizes this chapter.

3.2 The Way Point Routing (WPR) Hierarchy

Our intuition for the Way Point Routing hierarchy comes from a close comparison of DSR and AODV and from an attempt to make DSR scale to larger networks. Thus, we first present the comparison and then introduce the WPR hierarchy.

3.2.1 Comparison of AODV and DSR

Although both DSR and AODV are reactive routing protocols, they work differently in a number of aspects. Das et al. [32] give a good performance comparison of DSR and AODV. Below are the major differences between DSR and AODV:

1. The most noticeable distinction between DSR and AODV is that the packet-header overhead in DSR is larger than in AODV, because data and control packets in DSR typically carry complete route information. The header overhead restricts the scalability of DSR.

2. DSR can learn more routing information from the traffic than AODV, because DSR packets contain complete route information. More learned routes generally result in fewer route discoveries in DSR.

3. DSR is potentially able to obtain multiple routes through multiple route replies. In DSR, the destination responds to all valid received route requests (RREQs), but in AODV the destination only

responds to the first valid received RREQ. Pearlman [78] show that in multiple channel networks, multiple path routing can significantly improve the performance.

4. Compared to DSR, AODV is able to run on larger networks. As observed in our simulations (Section 3.4), DSR delivers 51% of all data packets for networks with 600 nodes in an area of $2800 \times 2800 m^2$, whereas AODV delivers 89% of all data packets for the same configuration. AODV maintains route information in a distributed manner — every intermediate node along a route keeps a forwarding pointer (next hop) towards the destination, whereas DSR maintains route information only at source nodes. Maintaining information in distributed manner is more suitable for MANETs.

5. If security is a major concern, DSR is easier to secure than AODV. In DSR, the source designates the path to be used to route data packets. Therefore, DSR can select a secure route that consists of nodes it trusts. A large amount of work on secure routing in MANETs is based on DSR [22, 55, 71, 76]. Gupte [48] and Hu [54] present surveys on security issues in MANETs.

In DSR, source routes carried in data packets are likely to cause significant overhead in larger networks where routes are longer. The Internet draft of DSR [60] recommends that DSR is suitable for MANETs "with up to around two hundred nodes". In both DSR and AODV, when a route breaks due to node mobility or node failure, the routing protocol discards the broken route and discovers a new route from the source to the destination. This approach has scalability and performance problems in larger networks. The local repair optimization to AODV is suitable for situations where link failures occur near the destination.

The problems mentioned above are inherent with flat routing protocols. Flat routing protocols maintain a route at the grain of the whole path, i.e., from the source to the destination. This approach works well for small networks, where routes are typically short. When the network grows, routes become longer and break more often. If routing protocols still maintain routes at the grain of the whole path, the routing overhead is high, because routing protocols discover and discard new routes frequently.

3.2.2 The Way Point Routing Approach

We use $manageable\ grain$ to describe the maintenance of a route. The manageable grain is *coarse* if a routing protocol maintains routes at the grain of whole path from source to destination. Flat routing protocols, such as AODV and DSR, maintain routes at the coarse manageable grain, and they discard a route if one hop on the route is broken. Generally, local repair is not applicable because no finer manageable grain is available.

A finer manageable grain is desirable when maintaining routes in larger networks where routes are typically longer. Our approach achieves a finer manageable grain by using a hierarchy of nodes on routes. In the hierarchy, we divide a route into several sub-routes, which we call $segments$. After dividing the route into segments, we obtain a two-level hierarchical routing model: the global inter-segment routing and the local intra-segment routing. At the higher level, we use a global inter-segment routing protocol from the source to the destination; at the lower level, we use a local intra-segment routing protocol within each segment. If needed, we can establish a hierarchy with more than two levels by further dividing the segments. We use the two-level hierarchy in our discussion. In our hierarchical routing model, the manageable grain is a segment. Managing routes at the segment grain makes it possible to conduct route maintenance activities locally, typically on a segment.

We divide a route into segments by selecting a number of *waypoint nodes* from the route. Therefore, we call our hierarchical routing model Way Point Routing (WPR). The source and the destination are also waypoint nodes. We call other nodes on the route *forwarding nodes*. Each segment starts with a waypoint node called the *start node* and ends with a waypoint node called the *end node*. The start and end nodes of a segment are generally connected by a number of forwarding nodes. Two neighboring segments share a common waypoint node, which acts as the end node of the upstream segment and the start node of the downstream segment. The WPR approach has the following advantages:

- First, since WPR maintains routes at a finer grain (i.e., a segment), WPR can fix a broken route locally at the level of a segment. When a route breaks, usually only a few hops are broken but other hops are still intact. Fixing a broken route within a segment extends the lifetime of the route and minimizes expensive, time-consuming global route discoveries. Thus, WPR substantially reduces the routing overhead and improves routing performance.

- Second, the length (hop count) of each segment on a route can be different, and the length of segments on different routes can be different. These flexibilities make WPR an adaptive routing scheme, which is important for MANETs where various network scenarios exist. For example, if a network is stable and nodes move slowly, WPR can use longer segments to reduce the overhead of maintaining the hierarchy; if nodes move faster, WPR can use shorter segments to facilitate route repairs.

- Finally, the hierarchy in WPR only involves the nodes on active routes, whereas other nodes save their resources.

Figure 3.2 gives an example of how WPR divides a route from node A to node J into segments. We select nodes D and G as waypoint nodes. Source A and destination J are also waypoint nodes. Waypoint nodes D and G divide the route into three segments: $A - b - c - D$, $D - e - f - G$, and $G - h - i - J$. To distinguish waypoint nodes from forwarding nodes, we use uppercase letters to label waypoint nodes and lowercase letters to label forwarding nodes. The nodes between the start and the end nodes of a segment are forwarding nodes. A segment can be simply identified by its start and end nodes. For example, we refer to segment $A - b - c - D$ as segment AD.

3.2.3 Combining AODV and DSR in WPR

For the issue of what protocols are suitable for inter-segment routing and intra-segment routing in WPR, theoretically we can use many existing routing protocols. In our instantiation, we use DSR for global inter-segment routing and use AODV for local intra-segment routing, and we term this instantiation the DSR Over AODV (DOA) routing protocol. Our choice of these protocols is motivated by the following reasons:

1. Researchers generally consider DSR and AODV as the two standard on-demand routing protocols. By combining them hierarchically, we expect DOA to inherit the strengths of both DSR and AODV, and thus, exhibit better scalability and performance. It turns out that this combination indeed improves the scalability of DSR greatly and reduces the overhead of AODV significantly.

2. DOA makes it possible for DSR routing and AODV routing to coexist in the same network. In fact, DSR and AODV are two special cases of DOA. When the segment length is one, DOA becomes DSR; when the segment length is a large number, DOA becomes AODV.

3. For local intra-segment routing, we use AODV because of its efficient operation and its ability to run on larger networks (with a rapidly increasing routing overhead though). Using AODV as the intra-segment routing protocol allows longer segments to exist in WPR, and thus makes the segment division in WPR more flexible. When needed, WPR can use long segments to reduce the number of waypoint nodes on a route.

We now explain the operation of the DOA routing protocol in the next section.

3.3 The DOA Routing Protocol

In DOA, DSR runs at the inter-segment level, and the source route option in the headers of data packets contains only waypoint nodes. AODV runs at the intra-segment level. DOA works on an on-demand basis. In DSR, the use of promiscuous receive mode is optional, and some optimizations can take advantage of its availability. But the use of promiscuous mode increases CPU processing overhead and power consumption. We do not use promiscuous mode in our implementation. Like other on-demand routing protocols, we address four issues in DOA: route discovery, data forwarding, route maintenance, and loop handling. We define four types of control messages: route request (RREQ), route reply (RREP), route error (RERR), and route activate (RACT). We use the RACT message to set up a reverse path from the end node to the start node of a segment, and we will explain it at the end of Section 3.3.1 in detail. To distinguish the messages used for inter-segment routing and intra-segment routing, we add subscript *inter* and *intra* to messages (e.g., RREQ$_{inter}$ and RREQ$_{intra}$).

3.3.1 Route Discovery

Inter-segment route request

When a source node requires a new route to a destination node, it broadcasts a RREQ$_{inter}$ message. A RREQ$_{inter}$ is uniquely identified by the combination of the source address and the source's broadcast id number. As in DSR, the RREQ$_{inter}$ message records the list of nodes it has traversed. When an intermediate node receives RREQ$_{inter}$, it first determines whether this request is a duplicate by consulting its request-seen table. Nodes discard duplicate requests. If the RREQ$_{inter}$ is new and its TTL is greater than zero, the intermediate node appends its address to the path recorded in the RREQ$_{inter}$ and re-broadcasts the request to its neighbors. In DOA, intermediate nodes do not generate route-reply messages even if they have a route to the destination in their caches, because the routes provided by intermediate nodes may be outdated. When the RREQ$_{inter}$ message reaches the destination, the destination node replies to the request as described next.

Inter-segment route reply

In DSR, for one route discovery identified by a source address and a broadcast id, there is no limit on the number of RREPs that the destination can send to the source; the destination sends a RREP whenever it receives a RREQ. We observed in our simulations that most of the paths discovered by RREQ messages share a large number of common nodes; only the last few hops are different. Based on this observation, for one inter-segment route discovery, we use the parameter MAX_REPLY_TIMES to limit the number of replies that the destination can send to the source. We set MAX_REPLY_TIMES to 2 in our simulations.

If the length of the path recorded in RREQ$_{inter}$ exceeds a threshold value (DEFAULT_SEGMENT_LEN), the destination divides the path into segments by selecting waypoint nodes from the path. There are several ways to select waypoint nodes. A naive method is selecting waypoint nodes that divide the path into segments evenly, i.e., the length (hop count) of each segment is roughly equal to DEFAULT_SEGMENT_LEN. For example, suppose the path from a source A to a destination J is

$$A - B - C - D - E - F - G - H - I - J,$$

and suppose the value of DEFAULT_SEGMENT_LEN is 3. A possible path division would be

$$A - b - c - D - e - f - G - h - i - J,$$

which contains three segments: AD, DG, and GJ. The length of each segment is equal to three. We can also select waypoint nodes according to other criteria, such as velocity or security association. Using lower-velocity nodes as waypoint nodes apparently makes routes last longer; using nodes from the same security association makes routes more secure. In our current implementation, we use the naive method to divide a path. The parameter DEFAULT_SEGMENT_LEN determines how many segments DOA divides a route into.

The destination generates a RREP$_{inter}$ message after it has divided the path. The destination then places the path that includes both waypoint nodes and forwarding nodes into the RREP$_{inter}$. Using our example, $b - c - D - e - f - G - h - i - J$ is placed into the RREP$_{inter}$. The path does not explicitly list the source address because it appears as the destination of the RREP$_{inter}$ message. Intermediate nodes use this detailed path to establish the route and relay the RREP$_{inter}$ back to the source.

We need to add more information into the RREP$_{inter}$ for intra-segment routing. As in AODV, RREP$_{inter}$ carries the sequence number of the end node of the current segment and the hop count to that node. We denote these two fields as end_seq and $end\_hopcount$, respectively. They have the same purposes as in AODV: loop prevention and route comparison. In our example, since the current segment is the last segment and the end node is the destination, the destination sets the end_seq to its own sequence number, and sets the $end\_hopcount$ to 1.

The destination sends the RREP$_{inter}$ message to the source along the reverse path contained in RREP$_{inter}$. Nodes transmit RREP messages using unicast. Before explaining how intermediate nodes and the source node handle RREP$_{inter}$, we need to introduce two tables maintained by all nodes in the network: $route\ cache$ and $routing\ table$. DOA uses the route cache for inter-segment DSR routing and the routing table for intra-segment AODV routing.

An entry in the route cache has a $\langle destination : source\_route\rangle$ structure, as in DSR. The route cache stores source routes from the current node to destination nodes. A source route only contains waypoint nodes. Therefore, any two neighboring nodes on a source route in DOA are one segment away from each other, instead of one hop away as in DSR. Typically, one segment consists of multiple hops.

An entry in the routing table uses a $\langle end\_node : next\_hop\rangle$ structure, as in AODV. An end node corresponds to a destination in AODV. The routing table stores the next hop to the end node of a segment, where the current node is either a forwarding node or the start node of this segment. The difference between the AODV routing table and the DOA routing table is that the destination field in the AODV routing table is the final destination of data packets, whereas in DOA, it is the end node of a segment and not necessarily the final destination.

When an intermediate node receives the RREP$_{inter}$, it first determines whether it is a waypoint node or a forwarding node on the replied route. If it is a waypoint node, it updates both its route cache and its routing table. If it is a forwarding node, it only updates its routing table.

To update its route cache, the node inserts source routes to the destination and to downstream intermediate waypoint nodes into its route cache. In our previous example, the route is $A - b - c - D - e - f - G - h - i - J$. Upon receiving the RREP$_{inter}$, node G adds $\langle J : J\rangle$ to its route cache, node D adds $\langle G : G\rangle$ and $\langle J : GJ\rangle$ to its route cache, and so on.

To update its routing table, the node inserts or updates an entry in its routing table if the intra-segment route ($\langle end\_node : next\_hop \rangle$) in the RREP$_{inter}$ is new or better. For the inserted or updated entry, the end node is the end node of the current segment, and the next hop is last node that relayed the RREP$_{inter}$ to the current node. Each entry in the routing table has a boolean field named $start\_node$, which is true only if the current node is the start node of the current segment. The $start\_node$ field determines whether to forward an intra-segment route error further, which we explain in detail in Section 3.3.3.

Using the previous example where the replied route is $A - b - c - D - e - f - G - h - i - J$, when node h receives the RREP$_{inter}$, it searches for the route to end node J in its routing table. If h finds no route or the found route is old, h establishes a new route to J. For the new route, the next hop is i, and the hop count is 2.

When the RREP$_{inter}$ arrives at an intermediate waypoint node, it means that the RREP$_{inter}$ is leaving a segment where the current node is the start node and entering an upstream segment where the current node is the end node. Therefore, this node sets the end_seq in the RREP$_{inter}$ to its own sequence number and resets the $end\_hopcount$ in the RREP$_{inter}$ to 1. Each node maintains a sequence number as in AODV.

Like AODV, the intra-segment level of DOA needs a reverse path from the end node to the start node of a segment. In order to let the end node and forwarding nodes on the segment know the sequence number of and the hop count to the start node, the start node sends a *route activate* (RACT) message to the end node. The RACT contains the two values, denoted as $start\_seq$ and $start\_hopcount$ respectively. The start node sets the $start\_seq$ to the sequence number of the start node, and sets the initial value of $start\_hopcount$ to 1. Forwarding nodes increase the $start\_hopcount$ when they relay the RACT. DOA uses RACT because when nodes propagate RREQ$_{inter}$s, although downstream nodes could learn the list of upstream nodes carried by RREQ$_{inter}$s, the route has not been divided into segments yet.

When the RREP$_{inter}$ reaches the source node that initiated the inter-segment route discovery, the source updates both its route cache and its routing table as described above. Now the source is ready to transmit data packets as described next.

3.3.2 Data Forwarding

To transmit a data packet, the source node first sets the source route option in the data packet for inter-segment routing. The source gets a source route to the destination from its route cache, and inserts the source route into the header of the data packet, as in DSR. The difference is that in DOA a source route only contains waypoint nodes.

After setting the source route option for the data packet, the source node processes intra-segment routing for the first segment on the route. It gets the next hop to the end node of the first segment by consulting its routing table, as in AODV. The difference is that AODV gets the next hop to the final destination, whereas DOA gets the next hop to the end node of the current segment. Next, the source node sends the packet to the next hop, which typically is a forwarding node.

The source-route option of a data packet contains a field named $current\_seg$, which is the index of the current segment on the source route. This index lets intermediate nodes on the route know which segment the data packet is being routed in. Waypoint nodes maintain the value of $current\_seg$. When the data packet leaves an upstream segment and enters a downstream segment at a waypoint node, the waypoint node increases $current\_seg$ by one. This operation belongs to inter-segment routing. For intra-segment routing, intermediate nodes get the next hop to the end node of the current segment from their routing tables and send the data packet to that next hop. Inter-segment routing (only at waypoint nodes) and intra-segment routing (at both waypoint and forwarding nodes) continue until the data packet reaches the destination.

3.3.3 Route Maintenance

Due to the dynamic nature of MANETs, links on a route may fail. Route maintenance is the mechanism to handle route breaks. A node can confirm if a packet is correctly received by the downstream node through any of the three types of acknowledgments: link-level, passive (listening to the forwarding by next-hop node), and network-layer.

In flat reactive routing protocols, when a link failure occurs, the upstream node of the failed link typically sends a route error message to the source. The route error message tears down the old route, and the source may start another route discovery. Route maintenance in this manner creates a significant overhead of route requests and even causes the broadcast storm problem identified by Tseng *et al.* [100]. On the other hand, a route in DOA is composed of segments. Thus, DOA can repair a broken route locally, in the scope of the broken segment or in the scope of a few segments near the broken segment.

We define two levels of route repairs in DOA: intra-segment route repair and inter-segment route repair. When a segment is broken, DOA tries intra-segment route repair first. If intra-segment route repair succeeds, waypoint nodes on the source route do not change, and the source node does not need to be notified. If intra-segment route repair fails, DOA tries inter-segment route repair next, and this repair works over multiple segments, including the broken segment and downstream segments. The repair involves downstream instead of upstream segments because it is the start node of the broken segment that initiates the repair. If inter-segment route repair succeeds, the waypoint node that initiated the repair sends the repaired route to the source. The source uses the repaired route to transmit data packets thereafter. Otherwise, the waypoint node that initiated the repair sends an inter-segment route error message to the source, which subsequently may start another round of global route discovery as described in Section 3.3.1. The start node of the broken segment is the initiator of intra-segment and inter-segment repairs. In the following description, we simply denote the route repair initiator as the $initiator$, and denote the node that the initiator looks for (e.g., the end node of the broken segment) as the $target$.

Intra-segment route repair

Intra-segment route repair works in AODV mode. When a node finds the next-hop node is unreachable, it sends a RERR$_{intra}$ message to its precursor nodes, which use the current node as the next hop for some segments. Multiple segments will be broken if they all use the broken link as a hop. The RERR$_{intra}$ contains the broken link and the end nodes of the broken segments. Upon receiving the RERR$_{intra}$, a precursor node searches for the broken segments from its routing table and sets the state of the broken segments to $invalidated$. If the current node is not the start node for some of the broken segments (in its routing table, the $start\_node$ values for these segments are false), it relays the RERR$_{intra}$ toward the start nodes of all broken segments.

When the start node of a broken segment receives the intra-segment route error, it initiates the intra-segment route repair. The initiator broadcasts an intra-segment route request (RREQ$_{intra}$) looking for the end node (target of the route discovery) of the broken segment. The initiator sets the TTL of the RREQ$_{intra}$ to MAX_SEGMENT_LEN (its value is DEFAULT_SEGMENT_LEN + 1 in our simulations). Intermediate nodes that receive the RREQ$_{intra}$ forward the message if it is not a duplicate and its TTL value is greater than zero, and discard the message otherwise. When the target receives the first RREQ$_{intra}$ message, it sends a RREP$_{intra}$ message to the initiator. Intermediate nodes on the reverse path relay the RREP$_{intra}$ from the target to the initiator. Nodes transmit RREP$_{intra}$ messages using unicast. Intra-segment route repair succeeds when the initiator receives the RREP$_{intra}$.

Figure 3.3 shows an example in which the original path for segment DG is $D - e - f - G$ and the link between e and f breaks. Intra-segment route discovery succeeds, and the new path for the segment DG is $D - u - v - G$. During the period from initiating the intra-segment route repair to knowing the result of the repair, the initiator buffers data packets it receives that use the route under repair. If the intra-segment repair succeeds, the initiator transmits the buffered data packets; otherwise, it tries inter-segment repair and keeps buffering undeliverable data packets.

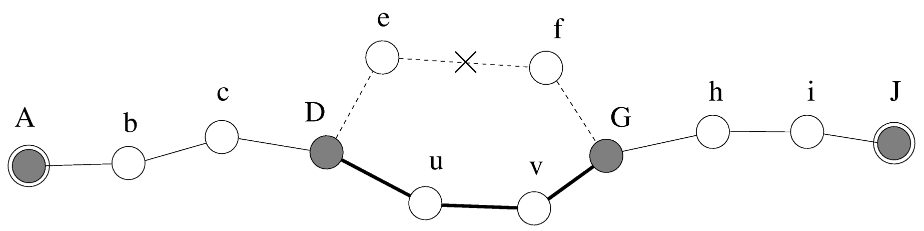


Figure 3.3: Intra-segment route repair.

Intra-segment route repair tries to establish a new path between the start and the end nodes of a broken segment. It may also establish a new path for the previous segment, even if the previous segment is not broken. The reason is that in AODV nodes learn new routes from route request messages: A node that receives a RREQ inserts or updates a route to the RREQ initiator. When the initiator broadcasts the RREQ$_{intra}$ message, the previous segment is in the range of the intra-segment route discovery. Figure 3.4 shows that an intra-segment route repair at the segment DG also builds a new path for the previous segment AD. The repair replaces the old segment $A - b - c - D$ with a new segment $A - w - E$. When a new path is discovered for the previous segment, it may be not activated. Therefore, the start node of the previous segment sends a RACT message to the initiator to activate the new path. The intra-segment route repair that happened at the broken segment also helps update the previous segment.

If intra-segment route repair succeeds (simulations show that in most cases it does), the waypoint

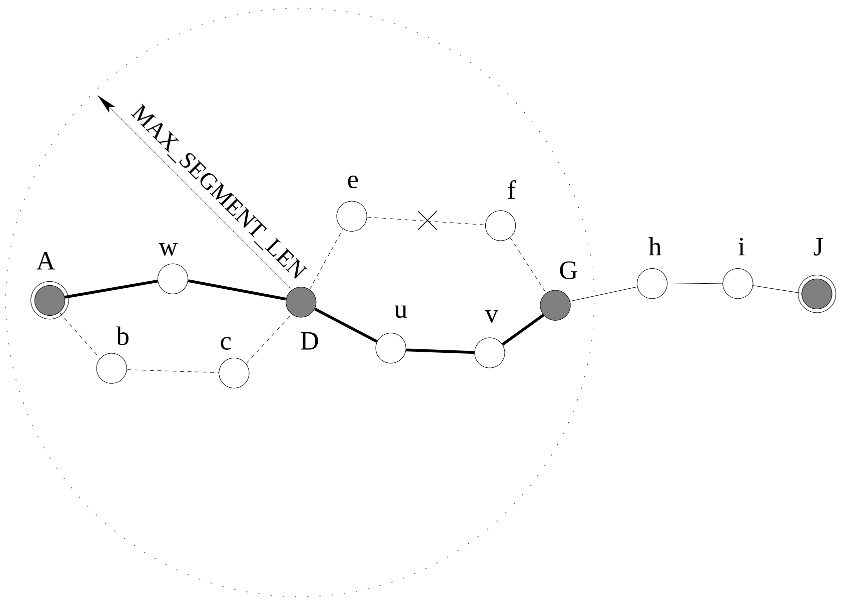


Figure 3.4: Intra-segment route repair updates the previous segment.

nodes on the repaired route do not change, and the source node does not need to be notified because data packets can be forwarded along the same waypoint nodes. If intra-segment route repair fails, the initiator tries inter-segment route repair, as described next.

Inter-segment route repair

The initiator discovers the failure of an intra-segment route repair by using a timer, called $Intra\_CheckReplied$, which the initiator sets when it initiates intra-segment route repair. If the timer expires, the initiator concludes that intra-segment route repair has failed and starts inter-segment route repair. Briefly, the process of inter-segment route repair includes discovering routes to downstream waypoint nodes on the route, repairing the broken route if the route discovery succeeds, and sending the repair result to the source.

Before explaining the detailed repair process, we introduce the definition of the inter-segment route error (RERR$_{inter}$) message in DOA. As described above, several types of information, such as whether the repair is successful or not, is required by upstream waypoint nodes. We utilize the RERR$_{inter}$ message to carry various information. An RERR$_{inter}$ message can be any of the following three types:

$REPAIR$: The route was broken but inter-segment route repair has succeeded. The message contains the repaired route.

$BROKEN$: The route was broken and inter-segment route repair has failed. The source node may need to start another global route discovery.

$LOOP$: The route has a loop. This type of RERR$_{inter}$ message is used for loop detection; we explain it in Section 3.3.4 in detail.

We denote these three types of RERR$_{inter}$ messages as RERR$_{inter}^{repair}$, RERR$_{inter}^{broken}$, and RERR$_{inter}^{loop}$, respectively.

As for the details of the inter-segment route repair, the initiator starts a localized inter-segment route discovery by broadcasting a RREQ$_{inter}$ message. The discovery process is similar to the global inter-segment route discovery discussed in Section 3.3.1. Whereas intra-segment route repair operates on the range of one segment (MAX_SEGMENT_LEN), inter-segment route repair operates on the range of two segments (2 * MAX_SEGMENT_LEN). If the broken segment is the last segment of the route, the target of route discovery is the destination.

Upon receiving the RREQ$_{inter}$ message, intermediate nodes handle the message as described in Section 3.3.1. They propagate the RREQ$_{inter}$ if it is not a duplicate and its TTL is greater than zero. When the target receives the RREQ$_{inter}$, it divides the path recorded in the RREQ$_{inter}$ into segments if the path is longer than DEFAULT_SEGMENT_LEN hops. The target sends an inter-segment route reply (RREP$_{inter}$) to the initiator. After receiving the RREP$_{inter}$, the initiator repairs the route by replacing the broken segment and those old downstream segments that are no longer used by the repaired route, with the new segments returned by the RREP$_{inter}$. Then the initiator updates its route cache, sends a RERR$_{inter}^{repair}$ message containing the repaired route to the source through upstream waypoint nodes and transmits buffered data packets using the repaired route. Upon receiving the RERR$_{inter}^{repair}$, upstream waypoint nodes update their route cache and transmit buffered data packets using the repaired route.

We illustrate the process of the inter-segment route repair by using the following example. The original route from the source A to the destination L is

$$A - b - c - D - e - f - G - h - i - J - k - L.$$

It can be written as $A - D - G - J - L$ at the inter-segment level. Suppose segment DG is broken, denoted as $A - D \times G - J - L$. Node D is the initiator of the route repair. D tries intra-segment route repair first to discover a path for segment DG, but the repair fails. Next, node D tries inter-segment route repair to discover a path to the end node J of the next segment GJ. Node D initiates a localized inter-segment route discovery, and the target is J. Suppose the route discovery succeeds and the new path from D to J is

$$D - u - v - W - x - y - J.$$

Initiator D selects node W as a waypoint node because the hop count from D to J is greater than DEFAULT_SEGMENT_LEN (its value is 3) and W divides the path from D to J into two segments. Initiator D repairs the original route; the repaired route is

$$A - b - c - D - u - v - W - x - y - J - k - L.$$

It can be written as $A - D - W - J - L$ at the inter-segment level. Node D updates its route cache, particularly, removing routes $D - G$, $D - G - J$ and $D - G - J - L$, and adding routes $D - W$, $D - W - J$, and $D - W - J - L$. Then D sends a RERR$_{inter}^{repair}$ message to source node A, informing that the inter-segment repair succeeded and the repaired route from A to L is $A - D - W - J - L$. Upstream waypoint nodes update their route caches similarly.

The initiator sets a timer called $Inter\_CheckReplied$ when initiating an inter-segment route repair. If the initiator receives no RREP$_{inter}$ when the timer expires, it concludes that inter-segment route repair has failed. The initiator then sends a RERR$_{inter}^{broken}$ to the source through upstream waypoint nodes, informing it that the route is broken and route repairs (intra-segment and inter-segment) have failed. Upstream waypoint nodes delete routes that use the broken segment from their route caches and delete data packets buffered during route repairs. The source node may start another global inter-segment route discovery to establish a route to the destination.

Multi-Target Route Discovery

During the process of an inter-segment route repair, the initiator conducts localized inter-segment route discovery to find a path to the end node of the next segment. End nodes of other downstream segments might be located in the discovery range for two reasons. First, this discovery covers the range of two segments, which is larger than an intra-segment route discovery, which covers one segment. Second, these nodes may move into range because of mobility. If these nodes send replies to the initiator, inter-segment route discovery will be more likely to succeed and may result in a shorter route (with fewer segments) than the original route. The potentially higher repair success rate is the motivation for multi-target route discovery.

In multi-target route discovery, the target field of the inter-segment route request (RREQ$_{inter}$) message may contain more than one node id, which means the initiator tries to find paths to multiple targets simultaneously. In our implementation, a RREQ$_{inter}$ message can contain up to three targets: the end nodes of the next two segments after the broken segment and the destination.

In our previous example, segment DG on route from A to L is broken ($A - D \times G - J - L$), and node D is the initiator for inter-segment route repair. When using multi-target route discovery, the RREQ$_{inter}$ message that D broadcasts contains J (the end node of the next segment) and L (the destination) in the target field. Both J and L can send a reply to D upon receiving the RREQ$_{inter}$ message.

3.3.4 Routing Loops Detection and Handling

Because local route repair operates within a limited range and lacks global knowledge, it may create routing loops. A routing loop is harmful for MANETs because it wastes resources such as bandwidth, processing power, battery, and memory. Therefore, absence of loops is one of the key requirements for all ad hoc routing protocols.

If a loop exists, the route passes an intermediate node more than once. We call this node the *loop node*. If a loop exists on a route found using DOA, it involves either one segment or multiple segments, which we call an *intra-segment loop* or an *inter-segment loop*, respectively. An intra-segment loop is not possible, because AODV is used for intra-segment routing, and its sequence number mechanism ensures the absence of loops.

For an inter-segment loop, the loop node belongs to at least two segments on a route. Three cases are possible:

$Case$ 1: The loop node appears more than once in a source route that contains only waypoint nodes. This type of loop is easy to detect, because DOA uses source routing at the global level. During inter-segment route discovery, an intermediate node does not handle an RREQ$_{inter}$ if it finds that its own address is already present in the recorded path. During data forwarding, the headers of data packets explicitly carry waypoint nodes, and DOA can easily detect duplicate waypoint nodes.

$Case$ 2: The loop node is a forwarding node of one segment on the route and is either a forwarding node or a start node of another segment on the route. This is a difficult situation and needs to be specially addressed.

$Case$ 3: The loop node is a forwarding node of one segment on the route and is also the destination of the route. This situation is possible when the destination moves towards the source and becomes a forwarding node of an upstream segment during route repair. This situation does not cause harm, because once the network layer finds that the destination address of data packets matches its own address, it stops routing the packets and sends them to the higher layer.

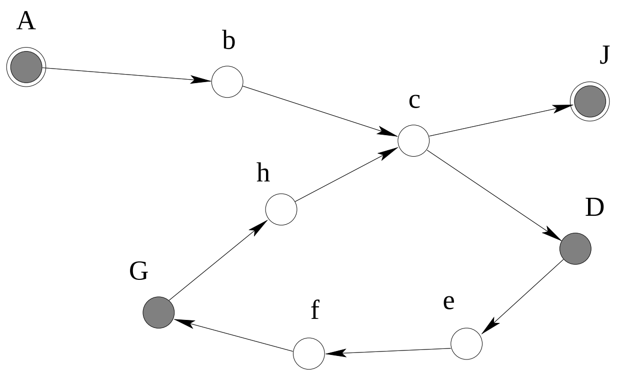
We designed a simple and efficient method to detect loops belonging to case 2 and case 3:

Loop Detection: *Upon receiving a data packet, an intermediate node checks its intra-segment routing table ($\langle end\_node : next\_hop \rangle$ structure). If it has routes ($next\_hop$) to the end node of the current segment as well as to the end node of another downstream segment, then a loop belonging to case 2 or case 3 exists.*

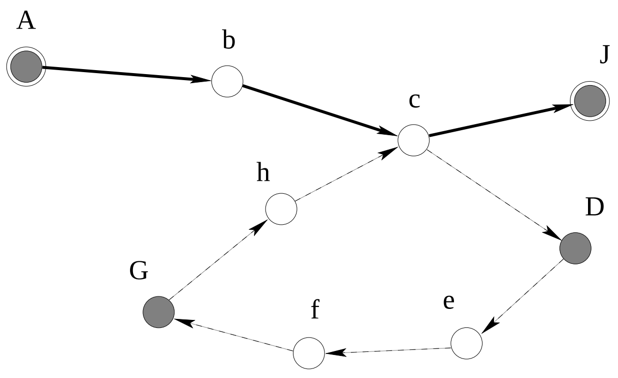
If a node finds a loop on a route, it removes the loop by changing the target (end node) of the intra-segment routing. It changes the target from the end node of the current segment to the end node of the downstream segment to which it knows the next hop. If the number of hops between the original start node and the new end node is larger than MAX_SEGMENT_LEN, the current node becomes a waypoint node on the route after it removes the loop.

Figure 3.5(a) shows a loop belonging to case 2. A path from source A to destination J is $A - b - c - D - e - f - G - h - c - J$. Node c is a forwarding node of both segment AD and segment GJ. On receiving the first data packet that uses this route, node c runs the loop detection routine and finds that its routing table has entries for end node D of the current segment AD as well as for end node J of segment GJ. Node c concludes that a loop exists. Node c removes the loop by changing the target of intra-segment routing from D to J. The new path becomes $A - b - c - J$, as shown in Figure 3.5(b). A loop belonging to case 3 can also be detected by the loop detection method. Suppose a path from source A to destination J is $A - b - c - D - e - j - G - h - i - J$. Node J is a forwarding node of segment DG and is also the destination. Node e detects the loop because it knows the next hop for end node G of the current segment and the next hop for end node J of segment GJ. Node e removes the loop and the new path becomes $A - b - c - D - e - J$.

If the node that removes a loop is not the source, it initiates an RERR$^{loop}_{inter}$ message, which contains information about the location of the loop and the new waypoint node if applicable. The node sends the message to the source through upstream nodes. Upon receiving the RERR$^{loop}_{inter}$,



(a) A loop is detected.



(b) The loop is removed.

Figure 3.5: Node c detects and removes a loop.

upstream waypoint nodes remove the loop from their route caches, and the source uses the modified route to transmit data packets thereafter.

3.3.5 Supporting Metrics Other Than Hop Count

Hop count is the most commonly used metric when routing protocols choose a path during a route discovery. However, as De Couto *et al.* [31] point out, the minimum hop count metric performs poorly in their 802.11b test-bed, because the hop count metric may fail to choose paths with high throughput. They propose a new metric called *expected transmission count* (ETX) to solve the problem. Therefore, it is desirable for WPR to support metrics such as ETX in addition to hop count. We use ETX as a representative in the following discussion.

Inherently, ETX is suitable in a DSR-like routing protocol, because ETX needs to collect the metric of each link on the path. AODV would face problems when supporting ETX, because its route request messages do not record the traversed links. Nevertheless, WPR is able to support ETX in the following ways:

First, WPR is a routing model that uses a high-level inter-segment routing protocol and a low-level intra-segment routing protocol. Many existing routing protocols are eligible for WPR. To support ETX, the inter-segment routing can use DSR, and the intra-segment routing can use DSDV or DSR. ETX has been implemented for both DSDV and DSR in [31]. Thus, a global route discovery can find a least-ETX path from a source to a destination, and a local route repair can find a least-ETX path between two waypoint nodes.

Second, in the case of DOA, global route discovery and inter-segment route repair are able to support ETX easily, because they work in source routing manner. The only problem is in intra-segment route repair, which conducts an AODV-style local route discovery. Then the problem becomes whether AODV can support ETX. We have designed a mechanism for AODV to use ETX-like metrics in [20] and discuss the details in Chapter 7.

3.4 Performance Evaluation

3.4.1 Simulation Setup

We conduct extensive simulations using GloMoSim 2.03 [21] to evaluate the performance of DOA and compare it with AODV and DSR. GloMoSim is a scalable simulation environment for wireless network systems. GloMoSim uses the parallel discrete-event simulation capability provided by PARSEC [13].

The MAC layer protocol used in the simulations is the IEEE standard 802.11 Distributed Coordination Function (DCF) [7]. DCF uses Request-To-Send (RTS) and Clear-To-Send (CTS) control packets for unicast data transmissions and the unslotted Carrier Sense Multiple Access protocol with Collision Avoidance (CSMA/CA) [7] for broadcast data packets and RTS control packets. The simulations use the two ray propagation model [91], 2 Mb/sec radio bandwidth, and 250m radio range.

The traffic is constant bit rate (CBR). We randomly select the source and the destination of each CBR flow but do not allow the source to be the destination. Each flow does not change its source and destination for the lifetime of a simulation run. Each source transmits data packets at a rate of four 512-byte data packets per second. The mobility model is random waypoint with the speed ranging from 0 m/s to 10 m/s and a pause time of 30 seconds. Exceptions will be noted otherwise.

We conduct three sets of simulations to evaluate the performance of DOA. First, we study the scalability of DOA in networks with 100 to 1400 nodes. Second, we study the performance of DOA when it works in the DSR and AODV modes. The number of CBR flows is 20 in both the simulation sets. Finally, we study the performance of DOA when the number of flows increases from 20 to 60 in networks with 400 nodes. In the simulations, we collect data for five metrics, namely, control overhead, packet delivery ratio (PDR), end-to-end delay, average route length, and route repair success ratio. We evaluate the first three metrics in all simulation sets. We evaluate the average route length metric in the first and second sets, and the metric of route repair success ratio in the first set.

When simulating DSR, we do not enable route replies from intermediate nodes, because the performance would become worse, because the implementation of DSR in GlomoSim 2.03 does not have a timeout mechanism to eliminate stale routes from route caches. We do not compare DOA with CGSR, ZRP, and LANMAR because DOA incurs much less routing overhead than CGSR, ZRP, and LANMAR. The latter maintain hierarchies proactively for all nodes, whereas DOA maintains the hierarchy reactively and only for nodes on active routes.

Each data point in the graphs is averaged over 10 simulation runs, each with a different seed. Error bars represent the 95% confidence interval for each data point. Each simulation lasts for 600 simulated seconds. Each source uses a warm-up time, randomly selected from 60 to 100 seconds before starting to transmit data packets, and ends transmitting data packets 20 seconds (cool-down time) before the simulation finishes. Most of the time, the flows in our simulations ran concurrently. We set the system parameters MAX_REPLY_TIMES, DEFAULT_SEGMENT_LEN, and MAX_SEGMENT_LEN to 2, 3, and 4, respectively.

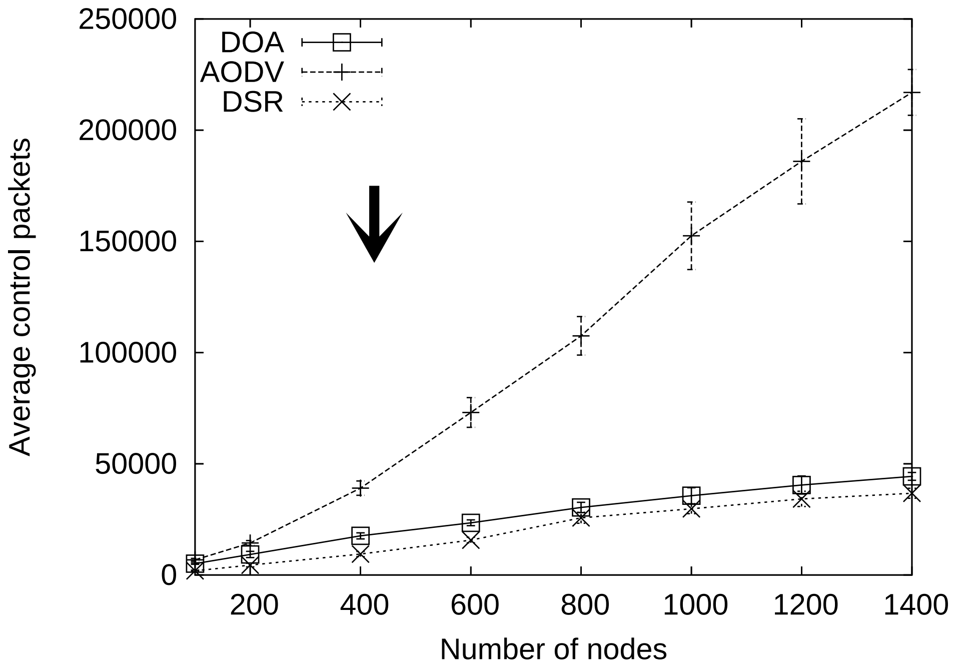
3.4.2 Results and Analysis

Scalability Study

In this set of simulations, we study the performance of DOA when the network size varies from 100 nodes to 1400 nodes. The network sizes and the respective network areas are shown in Table 3.1. We select the size and the area such that the node density is approximately constant, to properly evaluate the scalability of routing protocols. For each performance metric, we compare DOA with DSR and AODV. Figures 3.6(a)-(e) show the results. In each case, the bold arrow shows the preferred value, low or high, for the depicted measure. When applicable, DOA's results are depicted with a square symbol.

Table 3.1: Number of nodes and simulation space.

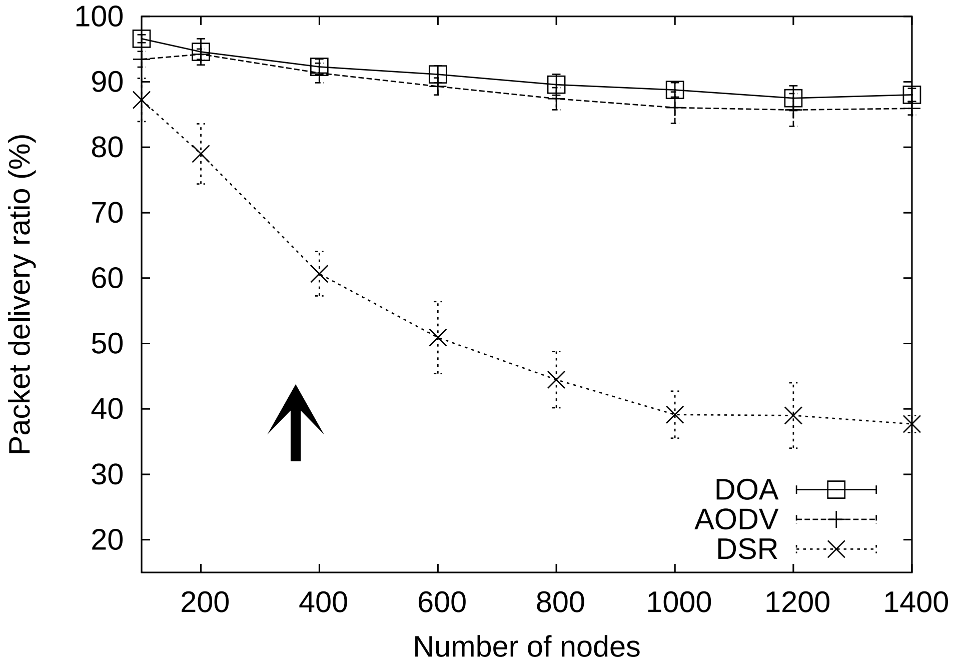
Network size	*Network area* (m^2)
100	1400×1400
200	2000×2000
400	2800×2800
600	3500×3500
800	4000×4000
1000	4500×4500
1200	4900×4900
1400	5300×5300



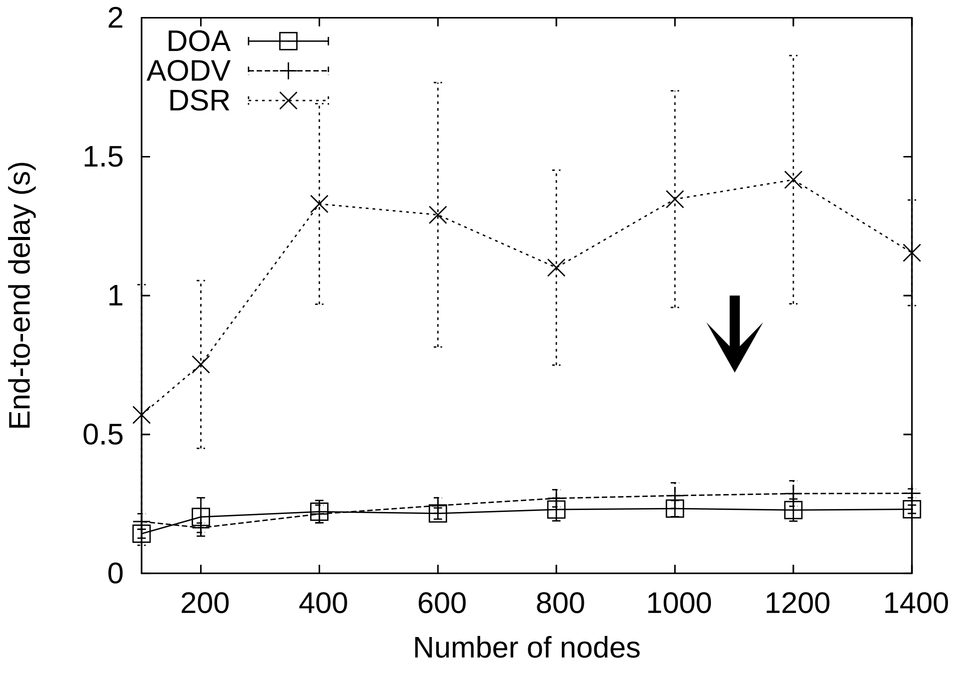
(a) Control overhead.

Figure 3.6: Scalability study.

Control Overhead. The main purpose of using a hierarchy in MANETs is to reduce the routing overhead. Figure 3.6(a) shows the control overhead of DOA, DSR, and AODV. We observe that DOA has much less control overhead than AODV when the network size increases, whereas supporting high PDR (Figure 3.6(b)). When the network size is 100 (nodes), three protocols incur

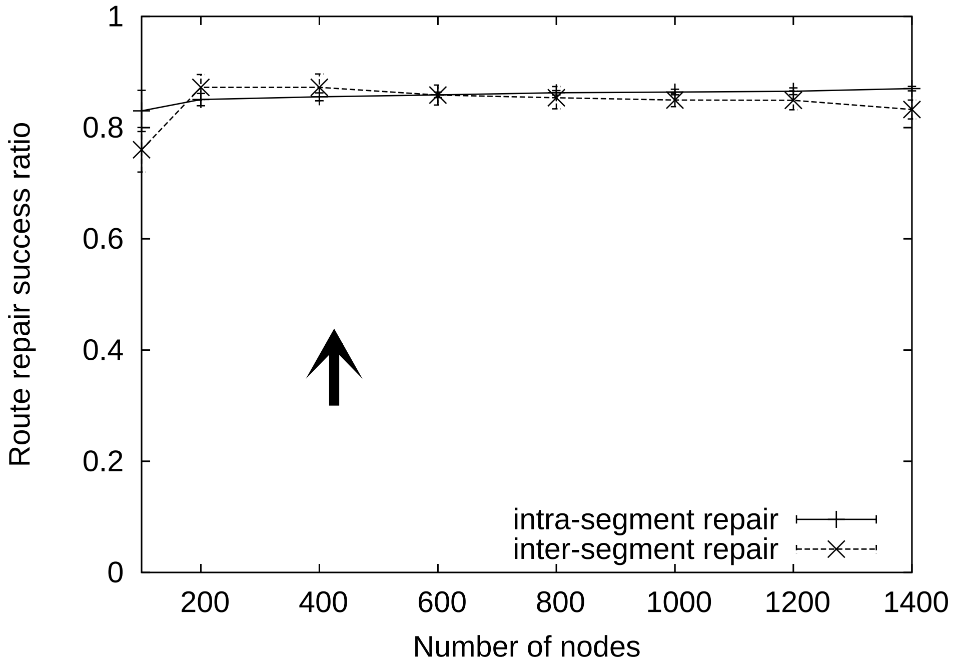


(b) Packet delivery ratio.

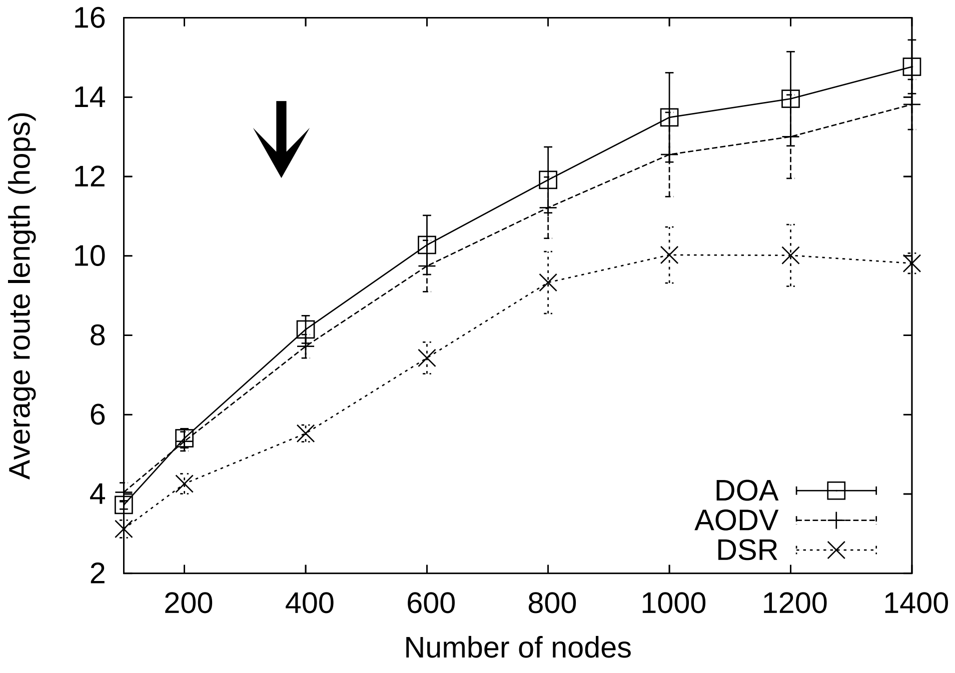


(c) End-to-end delay.

Figure 3.6: Scalability study (continued)



(d) Route repair success ratio.



(e) Average route length.

Figure 3.6: Scalability study (continued)

approximately the same amount of control overhead; when the network size is 600, DOA incurs 68% fewer control packets than AODV; when the network size is 1400, DOA incurs 80% fewer control packets than AODV. The larger the network, the more control packets are saved by DOA. DSR incurs slightly less control overhead than DOA, but the PDR of DOA is far better than DSR.

DOA significantly reduces control overhead because of its 2-level way point routing hierarchy. WPR localizes route maintenance activities and repairs a broken route either at the intra-segment level or at the inter-segment level. Low control overhead is critical for WPR to scale to large MANETs.

Packet Delivery Ratio. Figure 3.6(b) shows the PDRs of DOA, DSR, and AODV. We observe that DSR does not scale well to networks beyond a few hundred nodes. For networks with 600 nodes, DSR delivers 50% of all data packets. On the other hand, both DOA and AODV show high PDRs even for networks with more than 1000 nodes. For all network sizes from 100 nodes to 1400 nodes, DOA consistently delivers about 2-3% more data packets than AODV.

DOA maintains routes hierarchically and repairs a broken route locally. Thus, DOA improves the lifetime of routes and the packet delivery ratio. AODV shows very comparable PDR to DOA. However, the control overhead of AODV is significantly higher. DSR does not scale well because it has higher packet header overhead and keeps routing information in a non-distributed manner, as discussed in Section 3.2.1.

End-to-End Delay. Figure 3.6(c) shows the average end-to-end delay of DOA, DSR, and AODV. DOA exhibits the lowest end-to-end delay most of the time. The end-to-end delay of AODV is very comparable to DOA, whereas DSR has much higher end-to-end delay than the other two protocols.

In DOA, we expect a short end-to-end delay because the route repair mechanism recovers a broken route quickly, and data packets do not have to wait for another round of route discovery. However, DOA may not reduce the end-to-end delay dramatically for the following reasons. Intermediate waypoint nodes buffer data packets when they repair a broken route, and the buffer time contributes to the end-to-end delay. In the worst case, if route repair fails, the repair time is wasted.

The end-to-end delay in the figure does not show an obvious trend. The main reason for that is we used constant number of CBR sources. In small networks, the average length of routes is short but the networks are congested; in large networks with the same node density, the average length of routes is longer but the networks are less congested.

Route Repair Success Ratio. Figure 3.6(d) shows the success ratios of intra-segment route repairs and inter-segment route repairs in DOA. We observe that both success ratios are very high. More than 80% of intra-segment and inter-segment route repairs succeed for all network sizes. These results show that both intra-segment and inter-segment route repairs are worthwhile. The results also supports our suggestion at the beginning of the chapter that when a route breaks, usually only a few hops are broken but other hops are still intact. By repairing the broken route locally, we extend the lifetime of the route and reduce the number of global route discoveries.

Average Route Length. One concern for using local route repairs is that the repaired route may be sub-optimal. In the worst case, it may contain a routing loop. In DOA, a route cannot become very sub-optimal for the following reasons. First, if a forwarding node moves away, it causes a link break, and intra-segment route repair eliminates it from the current segment. The new route between the start and the end nodes of the broken segment becomes optimal again after the repair. Second, if a waypoint node moves away, it causes a segment breakage, and intra-segment route repairs may increase the hop count of the segment. But eventually, intra-segment route repair fails in the TTL range of MAX_SEGMENT_LEN hops. Then DOA tries inter-segment route repair, which usually eliminates the leaving waypoint node from the route. Finally, our loop detection mechanism effectively detects loops on active routes and removes them.

Figure 3.6(e) supports our analysis. The average route lengths of DOA and AODV are very comparable. Even for networks with 1400 nodes, the average route length of DOA (14.77 hops) is only 0.95 hop longer than that of AODV (13.82 hops).

The above results (Figures 3.6(a)-(e)) show that DOA indeed incorporates good aspects of DSR and AODV, and exhibits promising performance, especially in medium to large networks. Particularly, DOA substantially reduces the control overhead compared to AODV, and delivers significantly more data packets than DSR. At the same time, DOA maintains low end-to-end delay and short average route length after applying intra-segment and inter-segment route repairs. The repairs successfully recover a broken route most of the time.

DSR and AODV Modes

As mentioned at the beginning of the chapter, DSR and AODV are two special cases of DOA. When the segment length is one, DOA works in DSR mode; when the segment length is a large number, DOA works in AODV mode. This set of simulations aims to test how well DOA works in DSR and AODV modes, which we denote as DOA-DSR and DOA-AODV, respectively. We run the simulations in networks with 100 nodes in the area of $1400 \times 1400m^2$ and with pause time varying from 0 to 180 seconds. We set the segment length to 64 when simulating DOA in AODV mode. Other settings are the same as in the previous simulations. For each performance metric, we compare DOA-DSR and DOA-AODV with pure DSR and AODV. Figures 3.7(a)-(d) show the results.

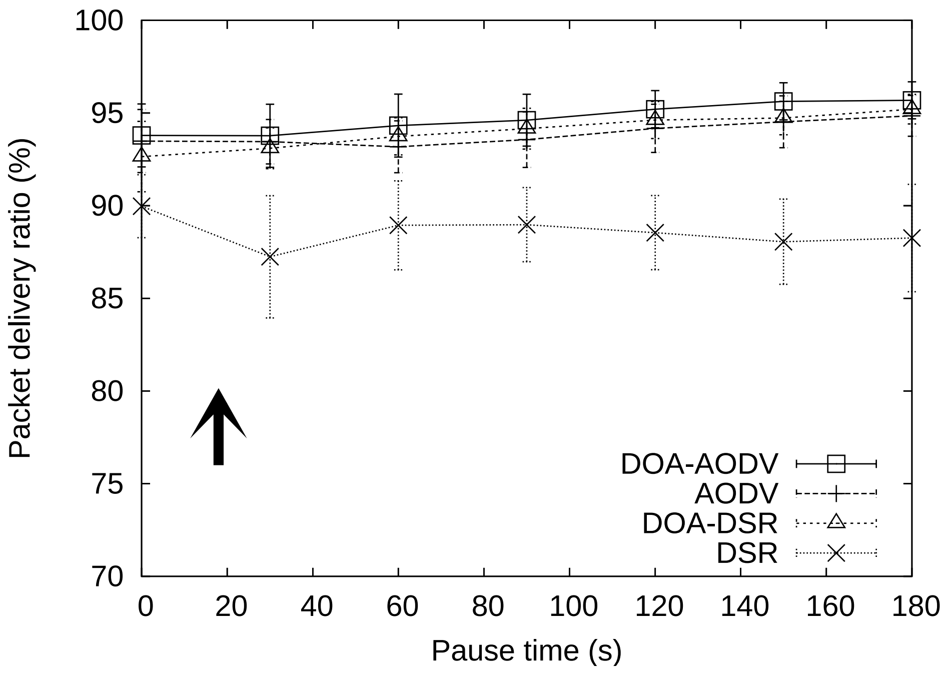
Figure 3.7(a) shows the PDRs of four protocols. We observe that DOA-DSR, DOA-AODV, and AODV all show high and very comparable PDRs (around 94%), whereas the PDR of DSR is lower (less than 90%). The improvement of DOA-DSR compared to DSR mainly comes from inter-segment route repairs.

Figure 3.7(b) shows the control overhead of four protocols. DOA-DSR and DOA-AODV incur more control overhead than DSR and AODV, respectively. This result is expected because DOA-DSR (DOA-AODV) needs more control messages than DSR (AODV) to maintain the hierarchy for active routes. DOA exhibits a performance advantage when the network size is large.

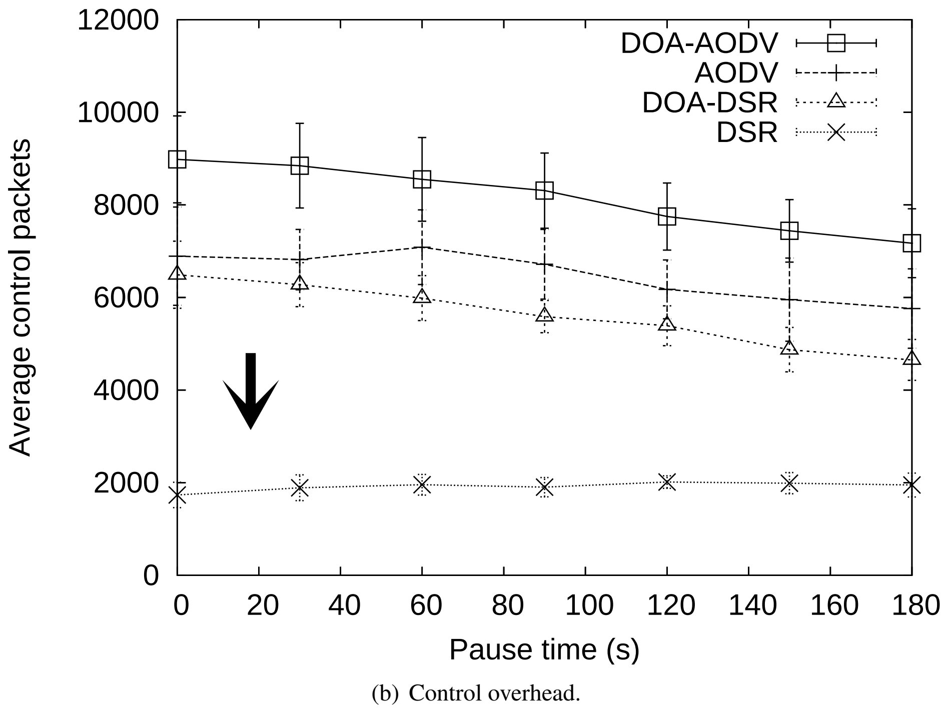
Figure 3.7(c) shows the end-to-end delays of four protocols. The end-to-end delays of DOA-DSR, DOA-AODV, and AODV are approximately the same (around 0.2 second), whereas the end-to-end delay of DSR is around 0.5 second, which is higher than other protocols. In DSR, the end-to-end delay is longer because routes break and need to be re-discovered more often, and data packets wait longer in buffers at source nodes for route re-discoveries. Buffer time contributes to the end-to-end delay.

Figure 3.7(d) shows the average route lengths of four protocols. The average route lengths of DOA-DSR, DOA-AODV, and AODV are approximately equal, whereas the average route length of DSR is about one hop less. DSR has shorter routes because routes break and need to be re-discovered more often, and routes are fresher and therefore better than in other protocols.

The above results (Figures 3.7(a)-(d)) show that the performance of DOA-AODV is comparable to pure AODV, whereas the performance of DOA-DSR is better than pure DSR in terms of packet

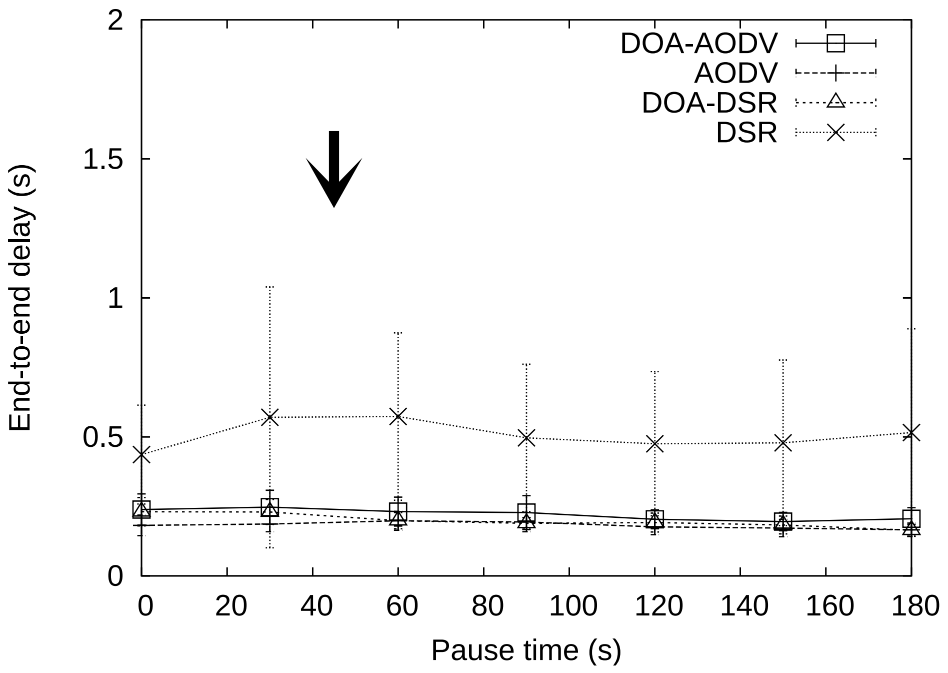


(a) Packet delivery ratio.

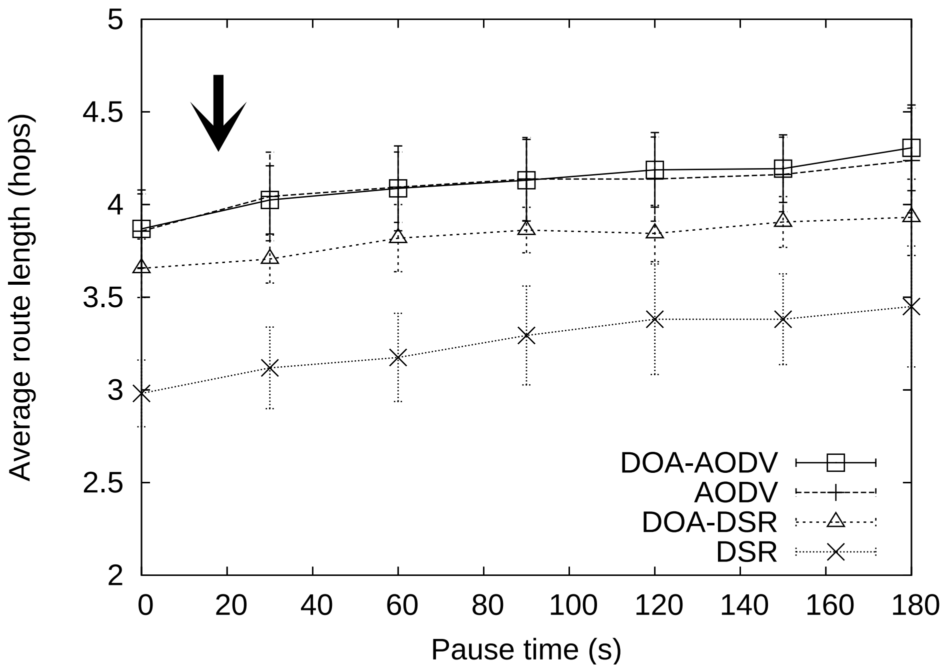


(b) Control overhead.

Figure 3.7: DOA in DSR and AODV modes.



(c) End-to-end delay.



(d) Average route length.

Figure 3.7: DOA in DSR and AODV modes (continued)

delivery ratio and end-to-end delay. The good performance of DOA in AODV and DSR modes is important, because it makes DOA very flexible: Applications that need AODV or DSR routing can coexist in networks using DOA.

Varying CBR Flows

This set of simulations aims to study the performance of DOA when the number of CBR flows increases. We run the simulations in networks with 400 nodes in the area of $2800 \times 2800m^2$. The number of CBR flows increases from 20 to 60. Other settings are the same as in previous simulations. For each performance metric, we compare DOA with DSR and AODV. Figures 3.8(a)-(c) show the results.

Figure 3.8(a) shows the PDRs of three protocols. The PDRs drop as the number of flows increases because the network becomes more congested. Nevertheless, DOA still shows the highest PDR. When the number of flows is 60, the PDR of DOA is 66%, whereas the PDRs of DSR and AODV are 52% and 34%, respectively. The PDR of DSR does not drop too much because DSR uses promiscuous listening mode, which we explain in more detail shortly.

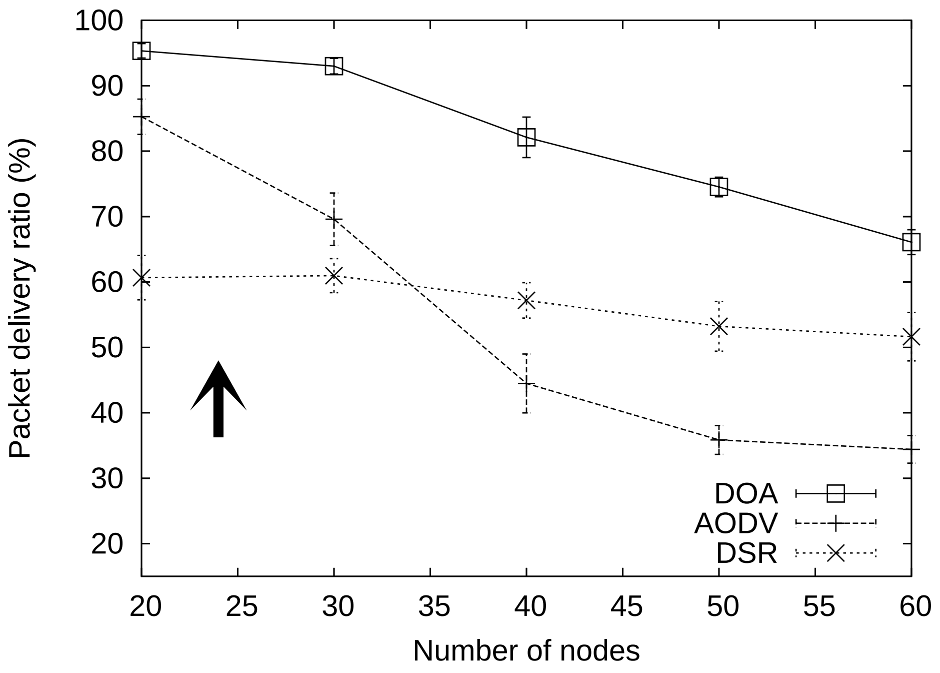
Figure 3.8(b) shows the control overhead of three protocols. We observe that DOA consistently saves more than 2×10^4 control packets per flow compared to AODV. However, when compared to DSR, DOA incurs more overhead. This result is mainly because DSR uses the promiscuous listening mode. Under this mode, when more traffic exists in the network, a node has more opportunities to learn new routes and thus conducts fewer route discoveries. We do not use the promiscuous listening mode in DOA because it would consume significantly more CPU and battery.

Figure 3.8(c) shows the end-to-end delays of three protocols. We observe that DOA incurs the lowest end-to-end delay among three protocols. The end-to-end delay of AODV is lower than DSR when the number of flows is less than 50 and becomes comparable to DSR after that.

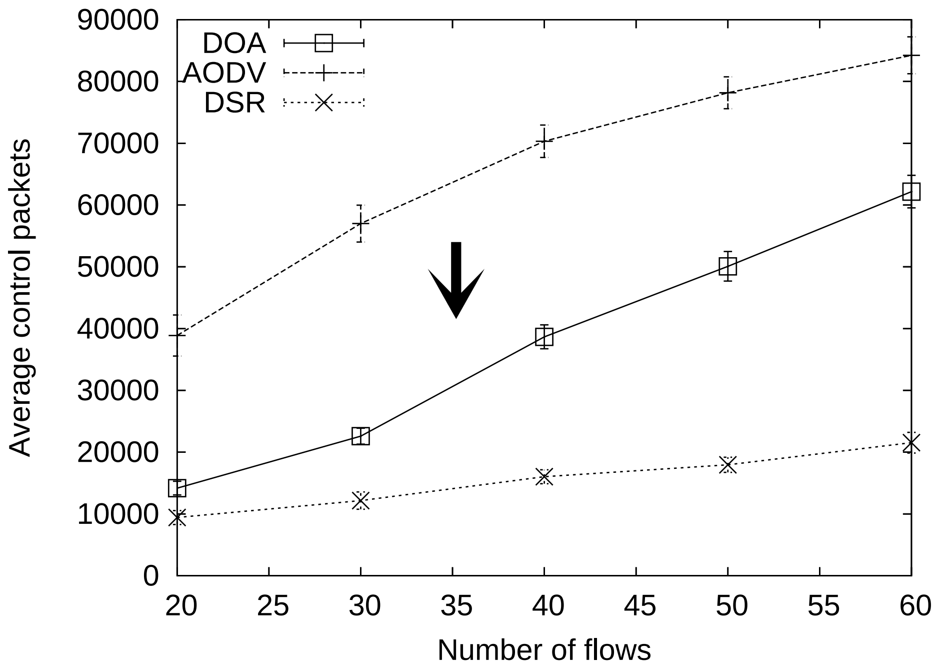
The above results (Figures 3.8(a)-(c)) show that DOA works well when the number of data flows increases in the network. DOA performs best among the three protocols in PDR and end-to-end delay. DSR incurs less control overhead at the cost of working in promiscuous mode.

3.5 Summary

We propose a new routing model, Way Point Routing (WPR), for Mobile Ad Hoc Networks (MANETs). WPR maintains active routes hierarchically. It selects a number of nodes on an active route as way-

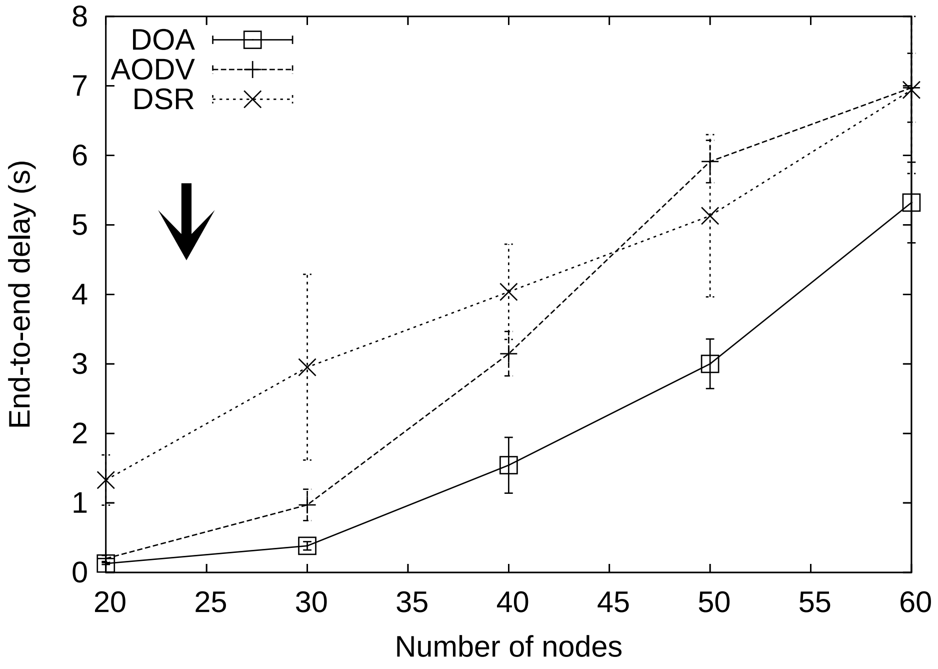


(a) Packet delivery ratio.



(b) Control overhead.

Figure 3.8: Varying CBR flows in 400-node networks.



(c) End-to-end delay.

Figure 3.8: Varying CBR flows in 400-node networks (continued)

point nodes and thus divides the route into segments. In WPR, an inter-segment routing protocol runs globally, while an intra-segment routing protocol runs locally. Therefore, WPR localizes route maintenance to one or a few neighboring segments. One distinct advantage of WPR is that when a route is broken because of node mobility or node failure, instead of discarding the whole route and discovering a new route from the source to the destination, only the two waypoint nodes of the broken segment need to find a new path. This approach has a clear performance advantage in routing overhead and end-to-end delay.

We present an implementation of WPR termed as DSR Over AODV (DOA). In DOA, we use DSR for inter-segment routing and AODV for intra-segment routing. DOA is the first work to combine DSR and AODV, the two standard on-demand routing protocols, in a hierarchical manner. We also present two novel techniques for route maintenance in DOA: a multi-target route discovery and an efficient loop detection method. We conduct extensive simulations to evaluate the performance of DOA. We compare DOA with AODV and DSR. Simulation results show that DOA scales well for networks with more than 1000 nodes, reduces routing overhead significantly, and performs better than or comparably to AODV and DSR in other metrics.

Chapter 4

SRR: Salvaging Route Reply

4.1 Introduction

In this chapter, we propose the idea of salvaging route reply (SRR) for on-demand routing protocols. Although DSR (dynamic source routing) [58] uses salvaging, it salvages data packets rather than route reply packets. We compare SRR with the salvaging in DSR in Section 4.3.5. In SRR, when an intermediate node cannot deliver a RREP to the intended next-hop node, it tries to salvage the RREP. We call this node the *salvor*.

We present two SRR schemes. In scheme one, the salvor actively broadcasts a salvage request message to its one-hop neighbors, asking for a cached path to the source. We propose scheme two as an improvement to scheme one. In scheme two, intermediate nodes passively maintain a backup path to the source utilizing duplicate RREQ packets. When the RREP cannot be relayed using the original path to the source, the salvor directly switches to the backup path. Therefore, SRR introduces very little extra overhead (in scheme one) or no extra overhead (in scheme two) to the routing protocol, but it has potential to substantially reduce the overhead of route discovery and significantly improve other performance metrics, such as packet delivery ratio and end-to-end delay.

We present the implementation of both SRR schemes in AODV (ad hoc on-demand distance vector) [85][1]. Additionally, we prove that in the implementation, routes are loop-free after salvage. We conduct extensive simulations to evaluate the performance of SRR in conjunction with AODV. The results show that SRR improves routing performance significantly over a wide range of system parameter values, measured by all critical metrics, including packet delivery ratio, control overhead, and end-to-end delay.

The rest of this chapter is organized as follows. In the next section, we discuss the motivation for SRR. In Section 4.3, we present two SRR schemes. Augmenting AODV with SRR is described in Section 4.4. Section 4.5 presents simulation setup and results. We summarize this chapter in Section 4.6.

[1]We also point out guidelines for the implementation of SRR in other on-demand routing protocols.

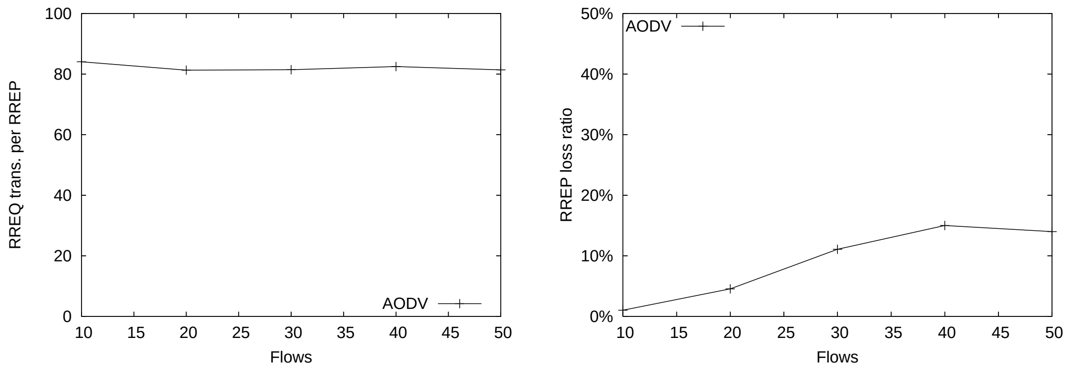
4.2 Motivations

In MANETs, a node may move from one location to another, turn its power on and off, join and leave the network, or completely fail. As a result, the network topology changes constantly and unpredictably. A link between two neighboring nodes may last for only a short time or temporarily become congested. It is a common incident that a node cannot successfully send a packet to its intended next-hop node. When a node cannot access the shared medium because the medium is continuously busy, or when a node cannot receive a link-layer or network-layer acknowledgment after transmitting a packet, the node concludes that it failed to send the packet. In most cases, the node discards the undeliverable packet. The packet loss impairs routing performance.

Among different packet losses, the loss of RREP packets causes the most serious impairment to the routing protocol. Usually, a RREP packet is obtained at the cost of flooding the entire network or a limited scope. In other words, the knowledge of a RREP message is obtained through tens, maybe hundreds of RREQ transmissions, an expensive and time-consuming process. Simply discarding an undeliverable RREP wastes a lot of route discovery effort. When a RREP is lost, the source node must initiate another round of route discovery, degrading routing performance. Therefore, in on-demand routing protocols, RREP packets should be given a higher priority than other packets. Unfortunately, to the best of our knowledge, no existing routing protocol attempts to salvage a RREP packet when it cannot be delivered to the intended next-hop node.

In order to illustrate the importance of RREP packets, we conducted a set of simulations, in which the routing protocol was AODV and the traffic was CBR (constant bit rate). The traffic load varied from 10 to 50 flows. Other setups are as described in Section 4.5. We only considered RREPs initiated by destinations[2]. Figure 4.1(a) shows the number of RREQ transmissions needed for destinations to initiate one RREP packet. On the average, each RREP is generated at the cost of more than 80 transmissions of the route request message. Figure 4.1(b) shows the loss ratio of RREP packets. In general, the loss ratio increases as the number of flows increases because the network becomes more congested. When the number of flows is 50, 14% of all RREP packets are lost. In the simulation, more than 1.6×10^5 transmissions of route request messages are wasted. These results justify salvaging undeliverable RREP packets.

[2]Intermediate nodes that have a route to the destination may answer a route request with a RREP message. We only salvage RREPs generated by destinations.



(a) RREQ transmissions per RREP (b) Loss Ratio of RREPs

Figure 4.1: The cost and loss ratio of RREP packets in AODV. Network area $1400 \times 1400 m^2$, simulation time $10min$, 100 nodes, 10 – 50 flows, maximum speed $20m/s$, and pause time $30s$.

We propose the idea of salvaging route reply (SRR) messages to minimize the loss of RREP packets and thus reduce the number of route discoveries. When a RREP is undeliverable due to link break or channel congestion, SRR tries to find an alternative path to relay the RREP to the source node, which initiated the route discovery. SRR is practical and applicable to all on-demand routing protocols. The direct advantage of SRR is a substantial saving of control overhead. Less overhead means less congestion. Less congestion results in shorter end-to-end delay and higher packet delivery ratio, because data packets travel faster in the network and experience less "overflow" (buffer is full) and "outdated" (packets wait too long) drops.

4.3 SRR: Salvaging Route Reply

In a route discovery, the destination sends a RREP message to the source after receiving the RREQ message. The RREP travels hop by hop towards the source, on the discovered route in reverse direction or on another route. When an intermediate node cannot deliver the RREP to the intended next-hop node because the next hop is unreachable, the intermediate node becomes a salvor and starts SRR. In this section, we first present two SRR schemes, then discuss several guidelines for SRR design. Moreover, we discuss MAC layer considerations for SRR and compare SRR with data packet salvaging in DSR.

4.3.1 SRR Scheme One (SRR1)

In SRR scheme one, the salvor runs local route discovery to find an alternative path to the source. The salvor starts the discovery by broadcasting a salvage request message (*SREQ*). The SREQ has a limited propagation scope. We use one hop, which means only one transmission for a SREQ message. Upon receiving the SREQ, if a neighboring node of the salvor has a route to the source or it is the source itself, it sends a salvage reply message (*SREP*) to the salvor. On receiving the SREP, the salvor sends the undeliverable RREP to the source using the alternative path returned by the SREP. The salvor sets a timer when starting the SRR route discovery. If no SREP returns when the timer expires, the SRR route discovery fails and the salvor discards the RREP without salvaging it.

A one-hop SRR route discovery will find an alternative path to the source most of the time for the following reasons. If the source is a one-hop neighbor of the salvor, it can send a SREP to the salvor directly. Otherwise, the one-hop neighbors of the salvor should have a route to the source in their routing tables, because when the original RREQ message is propagated outward from the source, intermediate nodes that relay the RREQ learn a route to the source from this message[3]. For example, in AODV, the last node from which the current node receives the RREQ is the next hop from current node to the source. In DSR, the reversal of the recorded path in a RREQ is the route from the current node to the source. Therefore, although the salvor does not have a path to the source since the old path is broken, it is highly likely that some of its neighbors have such a path and are able to answer the salvor's salvage request.

4.3.2 An Improvement: SRR Scheme Two (SRR2)

SRR1 is elegant because a salvor broadcasts a salvage request only once, and success is very likely. Unfortunately, SRR1 introduces extra control messages, SREQ and SREP. The extra message types increase the complexity of a routing protocol. In addition, salvaging adds a bit of delay to the route discovery process, because the salvor needs to send a SREQ and wait for a SREP. Of course, compared to the payoff, these shortcomings are nominal.

Therefore, we are motivated to improve SRR1 and develop SRR2, which requires no extra control message when salvaging an undeliverable RREP. Now the challenge is to find an alternative

[3]Bidirectional links are assumed. We discuss the situation of unidirectional links in Section 4.3.4.

path from the salvor to the source without sending any salvage request. We solve this problem by utilizing duplicate RREQ packets. In existing on-demand routing protocols, upon receiving RREQ packets for a route discovery, intermediate nodes typically only handle the first received RREQ and discard duplicate RREQs they receive later. The first or a duplicate RREQ can be distinguished because each RREQ message is identified, typically by the combination of a source id and a sequence number.

In SRR2, intermediate nodes still do not relay (broadcast) duplicate RREQs. However, intermediate nodes examine duplicate RREQs because they typically provide another path to the source. Intermediate nodes can store this alternative path to the source. Therefore, in addition to the primary path learned from the first RREQ, intermediate nodes also maintain a secondary path learned from duplicate RREQs. The secondary path serves as a backup route to the source in case the primary path is broken.

When relaying a RREP message, intermediate nodes first use the primary path. When the next hop on the primary path is unreachable, the intermediate node switches to the secondary path. Thus, the RREP that otherwise would be dropped is likely to be salvaged.

One question is how to choose the secondary path, because an intermediate node may receive multiple duplicate RREQs that represent multiple paths to the source. A node can select the secondary path based on different criteria. For example, the node may choose the path obtained from the second RREQ because it is the second fastest, or choose the path according to other metrics such as minimal hop count.

Single-Path and Multiple-Path Routing

Protocols such as AODV [85] are inherently single-path-based. It is necessary for SRR2 to check duplicate RREQs and learn a backup path to the source. Some protocols, such as DSR, are able to find multiple paths. So it may seem unnecessary to learn a backup path from duplicate RREQs, because a salvor may already have multiple paths to the source. However, when used without significant modifications, most of the paths discovered by DSR share a large number of common nodes; only the last few hops are different [15]. Therefore, we believe that SRR2 is useful for protocols such as DSR as well.

Maintaining a Backup Path

In on-demand routing protocols, nodes maintain a $request\_seen$ table to record route requests they have received. This record prevents nodes from forwarding duplicate requests received later for the same route discovery. We use the $request\_seen$ table to maintain backup paths. We make no change to the original routing table. Applying SRR2 to an existing routing protocol requires only minimal changes to the original protocol.

4.3.3 Guidelines for the SRR Design

Because SRR is an improvement to existing on-demand routing protocols, its operation should be simple and its overhead should be small. We give some guidelines for SRR design:

- Only allow a RREP packet to be salvaged at most once as it travels from the destination to the source. After a RREP is salvaged, the quality of the salvaged route from the destination to the source may be poor (not optimal or not stable). Additional link breakages on the route exacerbate its quality, so multiple salvaging efforts for such a route may not be worthwhile. Moreover, salvaging an undeliverable RREP packet only once helps prevent the RREP from entering a routing loop, in which different nodes salvage the RREP and send it to each other.

- Do not salvage RREPs generated by intermediate nodes (denoted as a RREP$_{in}$). First, an intermediate node generates a RREP$_{in}$ because it has a route to the destination. However, the route known by this node may be stale. Salvaging such a RREP$_{in}$ will only incur extra overhead. Second, a single RREQ may generate many RREP$_{in}$s, so the number of undeliverable RREP$_{in}$s may also be large. Salvaging them may introduce a large overhead to the routing protocol.

- Consider the issue of loop freedom. Loop freedom is a key requirement for ad hoc routing protocols because routing loops waste resources and may degrade routing performance seriously. After SRR salvages an undeliverable RREP, we must ensure that the route after salvage is loop-free. This issue is less serious in protocols that use source routing, in which packet headers carry routes and thus routing protocols can detect a loop easily. However, when used in some other protocols, e.g., protocols using a distance vector, we should carefully design the operation of SRR to prevent routing loops.

- In SRR1, salvage at most one RREP packet at a time in a given node. SRR route discovery transmits SREQ and SREP, increasing the traffic around the salvor. Multiple SRRs that run simultaneously at a salvor may cause congestion around the salvor and its neighbors.
- In SRR1, limit the scope of SRR route discovery. We recommend that it should only cover one-hop neighbors of the salvor, i.e., the initial TTL value of a SREQ should be set to one. A SREQ message then needs only one transmission.

4.3.4 MAC Layer Considerations

The majority of the routing protocols for MANETs assume bidirectional links: for example, AODV, TORA [77] and proactive protocols based on distance vector routing. The bidirectional links assumption facilitates implementing SRR. Otherwise, intermediate nodes may not be able to obtain a reverse path to the source from route request messages. Routing protocols require additional mechanisms to handle unidirectional links in the network.

Marina [70] shows the advantage of using unidirectional links is almost non-existent and proposes a *ReversePathSearch* technique to eliminate unidirectional links from route computations. Similar techniques include BlackListing [83] and Hello [56]. Such techniques guarantee that a reverse path from an intermediate node to a source still holds, because the forward route uses only bidirectional links.

If a routing protocol indeed needs to use unidirectional links, Duros et al. [36] propose a *tunneling* mechanism to emulate bidirectional connectivity upon unidirectional links. Thus the network layer protocols can still assume bidirectional links.

Another issue we need to consider in SRR is saving RREP packets being salvaged. When the MAC layer abandons transmitting a unicast packet, it notifies the routing protocol at the network layer. Some MAC layer protocols (or implementations) return the undeliverable packet as well. For example, the implementations of IEEE 802.11 MAC protocol [8] in simulators including GloMoSim [21] and ns-2 [38] hand the undeliverable packet back up to the network layer. In such cases, SRR can discover the information in an undeliverable RREP packet. Otherwise, SRR needs to save the RREP packet before the routing protocol hands it down to the MAC layer.

4.3.5 A Comparison to Packet Salvaging in DSR

In DSR, when a node fails to send a data packet to the intended next-hop node, in addition to sending a route error message to the source of the packet, it attempts to salvage the packet by searching in its own route cache for an alternate route to the destination. If such a route exists, the node makes necessary changes to the packet header and sends the packet using this route. Once DSR salvages a data packet, it marks the packet as *salvaged* to prevent the packet from being salvaged multiple times, which could lead to a routing loop because different nodes may salvage the packet and send it to each other.

SRR has two important differences with packet salvage in DSR. First, the packet salvaged is different. SRR salvages undeliverable RREP packets (control packets), but DSR salvages undeliverable data packets. RREP packets are more important than data packets for the performance of on-demand routing protocols. Second, the approach is different. SRR1 conducts a one-hop route discovery, and SRR2 utilizes duplicate RREQs to prepare a backup path to the source in advance, whereas salvage in DSR only searches the node's route cache. It neither sends out extra routing messages nor considers duplicate RREQs.

The differences between SRR and packet salvaging in DSR do not imply that SRR cannot be applied to certain on-demand routing protocols. In fact, SRR can be incorporated into all on-demand protocols in which the route discovery works through broadcasting a RREQ message and waiting for a RREP message.

4.4 Salvaging Route Reply in AODV

We now present the implementation of SRR1 and SRR2 in AODV [85]. We choose AODV because it is a prominent and widely used routing protocol. AODV is based on the traditional distance vector routing scheme, and it makes routing decisions on a hop-by-hop basis. AODV discovers a route by adding a backward routing entry pointing to the source at intermediate nodes when they propagate the RREQ message, and by adding a forward routing entry pointing to the destination at intermediate nodes when they relay the RREP message to the source. In the following discussion, when we mention the *original* route discovery, we mean the route discovery whose RREP SRR is salvaging. When we mention the *source*, we mean the source of the original route discovery.

During a route discovery, after the destination receives the RREQ message, it sends a RREP message to the source via intermediate nodes. If an intermediate node cannot deliver the RREP because the intended next-hop neighbor is unreachable, the node tries to salvage the RREP. First, the node checks two conditions. (i) The RREP is not generated by an intermediate node. (ii) The RREP has not been salvaged before (it is not marked as *salvaged*). If the RREP satisfies both conditions, the node tries to salvage the RREP using either SRR1 or SRR2.

Next, we discuss how to prevent a loop from being formed on the route after salvaging a RREP. Then, we present the design details of SRR1 and SRR2 in AODV. At the end of this section, we discuss the use of a new message type called a $route\ update$ message. We discuss loop prevention first because both SRR1 and SRR2 use it.

4.4.1 Loop Freedom

The salvor tries to find a new next hop to the source, either by broadcasting a SREQ message or by maintaining a backup path to the source when the salvor receives duplicate RREQs. In this subsection, we discuss the issue of loop freedom under SRR in AODV. We use the following notations in the discussion:

$Path(A,\ B),\ Path_{srr}(A,\ B)$:	Path from node A to node B before and after SRR, respectively.
$NextHop(A,\ B,\ \mathcal{P})$:	The next hop from node A to node B on path $\mathcal{P}$.
$HopCount(\mathcal{P})$:	The number of hops on path $\mathcal{P}$.
$S,\ D,\ X$:	Source, destination, and salvor nodes, respectively.

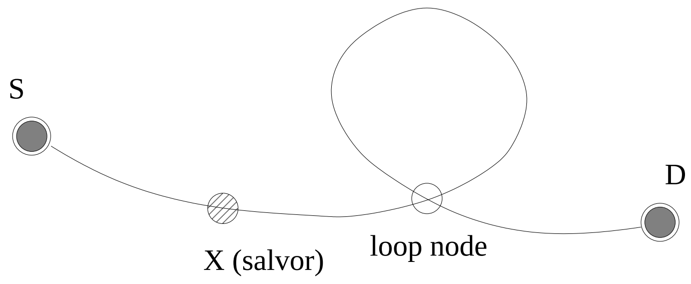
We define two conditions for preventing a loop from being formed on $Path_{srr}(X,\ S)$:

Condition 1: $NextHop(NextHop(X,\ S,\ Path_{srr}(X,\ S)),\ S,\ Path_{srr}(X,\ S)) \neq X$
Condition 2: $HopCount(Path_{srr}(X,\ S)) \leq HopCount(Path(X,\ S)) + 1$

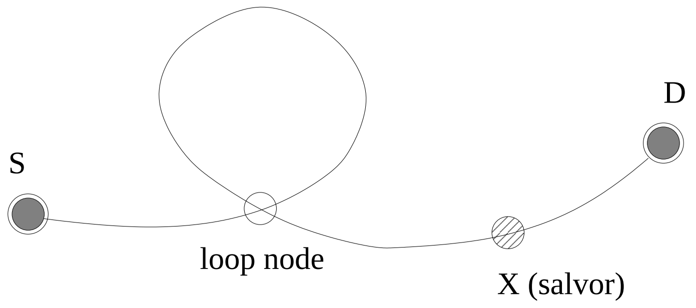
Condition 1 means the new next-hop node does not use the salvor as the next hop to the source. Condition 2 means the new path has at most one more hop than the broken path. We now prove that when the above two conditions hold, the route is loop-free after SRR salvages the RREP.

Theorem 1. *When both Conditions 1 and 2 hold, the route is loop-free after SRR salvages the RREP.*

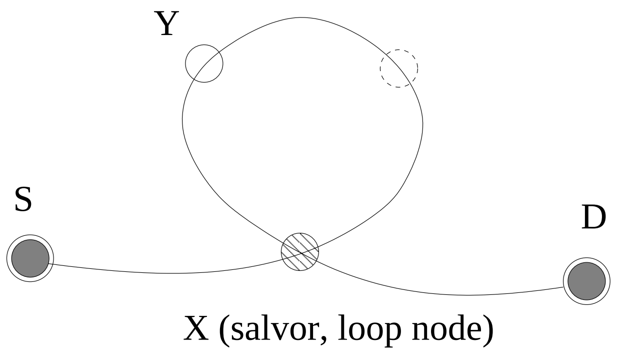
Proof. The proof is by contradiction. Assume that after salvage, the route contains a loop. We call the node that the route passes through twice the *loop node*. The order of the salvor (X) and the loop node on the route have three possibilities: (1) X is before the loop node; (2) X is after the loop node; (3) X is the loop node. Figure 4.2 (a), (b), and (c) show the three cases, respectively.



(a) Case 1: Salvor is before the loop node.



(b) Case 2: Salvor is after the loop node.



(c) Case 3: Salvor is the loop node.

Figure 4.2: Three possible orders of salvor and loop node if assuming a loop occurs after salvaging a RREP.

Cases (1) and (2) mean the loop node forwarded the RREQ message twice, which is not possible because in AODV, nodes forward the RREQ message for a route discovery at most once.

For case (3), besides the salvor X, there must be one or more nodes in the loop. When there are two nodes in the loop, the two nodes would be X and another node Y. Then, Y would use X as the next hop to the source, i.e., $NextHop(NextHop(X, S, Path_{srr}(X, S)), S, Path_{srr}(X, S)) = X$. This contradicts condition (i). When there are more than two nodes in the loop, the new path would have at least two more hops than the broken path, i.e., $HopCount(Path_{srr}(X,\ S)) > HopCount(Path(X,\ S)) + 1$. This contradicts condition (ii).

Therefore, the assumption that the route contains a loop is incorrect. The two conditions guarantee loop freedom for SRR in AODV. □

4.4.2 SRR Scheme One for AODV (AODV-SRR1)

In scheme one, if the node is not salvaging any other RREPs, it acts as a salvor and starts the SRR procedure. The salvor first saves the undeliverable RREP message. This will also prevent the salvor from salvaging other RREPs at the same time. Then, the salvor initiates a SRR route discovery by broadcasting a SREQ message. In the message, the *initiator* field is set to the salvor, the *target* field is set to the source (of the original route discovery), and the *TTL* field is set to one. The message also carries the hop count of the broken path (from the salvor to the source).

If the source receives the SREQ, the source sends a SREP to the salvor directly. Otherwise, the node that receives the SREQ consults its routing table for a route to the source. As discussed in Section 4.3.1, one-hop neighbors of the salvor are likely to have a route to the source because they recently propagated the original route request message and learned a reverse path to the source. If such a route exists, the node checks the two loop-prevention conditions discussed in Section 4.4.1. If both conditions are satisfied, the node responds to the salvor with a SREP message.

The salvor waits a short period of time (e.g., 2 * NODE_TRAVERSAL_TIME [83]) for the SREPs from its neighbors. Then it selects the best route from them according to the standard route update rules in AODV: use the route with newer sequence number; when sequence numbers are the same, use the route with the lowest hop count. Next, the salvor salvages the saved RREP by sending it to the source using the path just discovered. The salvor marks the RREP as *salvaged* to prevent other intermediate nodes from salvaging it again.

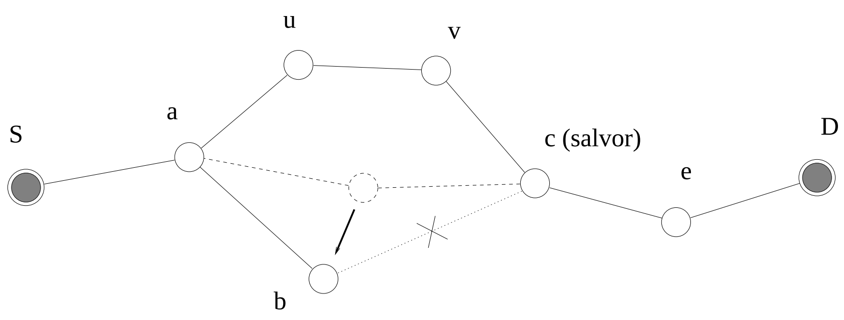


Figure 4.3: An example of SRR. Link $b - c$ is broken. Salvor node is c. Intended RREP return path is $D \rightarrow e \rightarrow c \rightarrow b \rightarrow a \rightarrow S$. Actual return path after SRR is $D \rightarrow e \rightarrow c \rightarrow v \rightarrow u \rightarrow a \rightarrow S$.

We give an example in Figure 4.3 to explain how AODV-SRR1 works. Node S is discovering a route to node D. D sends a RREP to S, and the original return path is $D \rightarrow e \rightarrow c \rightarrow b \rightarrow a \rightarrow S$. Node c cannot send the RREP to node b because b moves away. Node c becomes the salvor and broadcasts a SREQ. Node v receives the SREQ and has a route to S in its routing table, so v responds a SREP to c. c receives the SREP and successfully salvages the RREP by relaying it on the path discovered by SRR. Thus the return path after SRR is $D \rightarrow e \rightarrow c \rightarrow v \rightarrow u \rightarrow a \rightarrow S$.

4.4.3 SRR Scheme Two for AODV (AODV-SRR2)

In scheme two, during a route discovery process, intermediate nodes handle the first received RREQ packet as in the original AODV. When intermediate nodes receive a duplicate RREQ packet, which represents a new path to the source, they decide whether the new path can be used as a secondary path to the source. If a node does not have a secondary path yet, the node uses the new path as the secondary path if the new path satisfies the two loop-prevention conditions in Section 4.4.1. If the node already has a secondary path, it uses the new path if the new path has a lower hop count than the old path.

Later on, when an intermediate node cannot send the RREP to the next hop on the primary path, if the node has cached a secondary path to the source, the node marks the RREP as $salvaged$ and relays it using the secondary path. We illustrate how AODV-SRR2 works using Figure 4.3. For node c, the primary path to source S is $c \rightarrow b \rightarrow a \rightarrow S$ and the secondary path is $c \rightarrow v \rightarrow u \rightarrow a \rightarrow S$. When the primary path is broken, c salvages the RREP by relaying it using the secondary path.

4.4.4 Route Update Message (RUPD)

After the salvor salvages the undeliverable RREP using either AODV-SRR1 or AODV-SRR2, downstream nodes on the route (e.g., nodes e and D in Figure 4.3) may have incorrect route information to the source: for example, incorrect hop count to the source or incorrect sequence number for the source. This situation happens when the new path has different hop count from the broken path, or when downstream nodes have an outdated sequence number for the source. For the example in Figure 4.3, originally nodes e and D are 4 and 5 hops away from the source, respectively. But after the SRR, the distances become 5 hops and 6 hops. Therefore, SRR needs to notify downstream nodes the correct route information to the source.

We define a new message type for SRR, *route update (RUPD)*, to handle this situation. A RUPD is a unicast message that SRR uses to inform downstream nodes on the salvaged route about the source information: First, the sequence number (*SN*) of the source known by the salvor; second, the hop count to the source from the node that receives the RUPD. The following is the format of RUPD message:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Type  (RUPD)  |            Reserved             |   Hop Count   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Source Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Source Sequence Number                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

We illustrate how the RUPD message works using the example in Figure 4.3. After salvaging the RREP, the salvor c sends a RUPD message to node e. The RUPD contains the SN of the source known by c and a hop count of 5. When e receives the RUPD, it updates its routing table accordingly. Then, node e changes the hop count value in RUPD to 6 and sends it to node D. D processes the RUPD similarly but no longer relays it because it is the destination.

The RUPD message works well with AODV-SRR1, which has already introduced the SREQ and the SREP messages. However, for AODV-SRR2, the RUPD message is undesirable, because a design objective of SRR2 is to eliminate extra control messages. We solve this problem by using a RUPD IP option in the header of a data packet. The following is the format of a RUPD option:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |   Reserved    |   Hop Count   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Source Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Source Sequence Number                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The `Type` octet has 3 fields: `copy flag` (1 bit), `option class` (2 bits) and `option number` (5 bits). RFC 791 [87] defines and describes these fields. RFC 3692 [74] and RFC 4727 [39] give recommendations for assigning experimental and testing numbers when adding new functions using an IP option. In our simulations, we set `copy flag` to 1, `option class` to 0, and `option number` to 30. Thus, the value of `Type` is 158. Because the RUPD option uses 12 bytes, the value of `Length` octet is 12.

When adding a RUPD option to a data packet, we increase the IP header length (`IHL`, in 32-bit words) of the packet by 3, and append the RUPD option to the end of the IP header. To determine if a RUPD option exists in the IP header of a data packet, we check two conditions. First, the header length should be greater than 5, meaning the header carries one or more options. Second, the `Type` value of some option should be equal to the `Type` value of RUPD (158).

After the source receives a RREP marked as $salvaged$ and before the source uses the discovered route to send buffered data packets, the source inserts a RUPD option to the header of the first data packet to be sent. Upon receiving a data packet that contains a RUPD option in its header, intermediate nodes update their routing tables according to the RUPD option.

4.5 Performance Evaluation

We conduct extensive simulations to evaluate the performance of SRR. We simulate AODV-SRR1 and AODV-SRR2 and compare it with AODV. In this section, we first introduce the simulation setup and then present the results and analysis.

4.5.1 Simulation Setup

We conduct simulations using GloMoSim 2.03 [21]. The settings for radio model and MAC layer are the same as described in Chapter 3.4. The traffic is constant bit rate (CBR). We randomly select the source and the destination of each CBR flow, but do not allow the source to be the destination

Table 4.1: Setup summary of 100-node simulations.

Parameter	*Flows*	*Load* $(pkts/s)$	*Max Speed* (m/s)	*Pause Time* (s)
Flows	10 – 50	4	20	30
Load	20	4 – 14	20	30
Max Speed	30	4	5 – 30	0
Pause Time	30	4	20	0 – 50

(sources and destinations of different flows might coincide). Each simulation lasts for 1200 seconds. We do not change the source and the destination of a flow during the lifetime of a simulation run. After a simulation starts, each source waits a *warm-up* time, which we randomly select from 60 to 100 seconds, before sending data packets. The size of data packets is 512 bytes. Each source stops sending data packets 20 seconds before the simulation ended. We call this period the *cool-down* time.

The mobility model is random waypoint [23], which randomly chooses the speed of a node between a minimum and a maximum value. A node stays for a pause time after reaching a waypoint. A zero pause time means nodes move continuously. Unless specified otherwise, the following configurations for the traffic and the mobility are common among our simulations: CBR sending rate 4 $packets/s$, minimum node speed 0.1m/s, maximum node speed 20m/s, and pause time 30s.

In the performance evaluation, we first study AODV-SRR1, AODV-SRR2, and AODV in networks containing 100 nodes in an area of 1400×1400m^2. We organize the 100-node simulations into four groups by varying four network parameters: the number of CBR flows, packets sending rate at sources, maximum node moving speed, and node pause time, respectively. Table 4.1 summarizes the setups for these four groups of simulations.

We then studied AODV-SRR1, AODV-SRR2, and AODV in larger networks. The number of nodes was varied from 100 to 300, and the number of flows was 20% of the total number of nodes. For example, in a 300-node network, the number of flows was 60. To scale the simulations from small networks to larger networks, we maintained constant node density by increasing the network area accordingly. Table 4.2 gives the setups for this set of simulations.

In the performance study, we focus on three key metrics that researchers widely use in evaluating the performance of routing protocols: *packet delivery ratio (PDR)*, *control overhead*, and *end-to-end delay*. Moreover, we count SRR control packets, including SREQ, SREP, and RUPD, for

Table 4.2: Setup summary of simulations with varied network size.

Nodes	*Flows*	*Area (m^2)*
100	20	1400×1400
150	30	1700×1700
200	40	2000×2000
250	50	2200×2200
300	60	2450×2450

AODV-SRR1 in 100-node simulations with varied number of flows. This measure would reflect the advantage of SRR2 because compared to AODV-SRR1, AODV-SRR2 requires no extra control packets. Each data point in the graphs is averaged over 40 simulation runs, each with a different seed. Error bars represent the 95% confidence interval for each data point. We add a "goodness arrow" to each graph showing which direction is better.

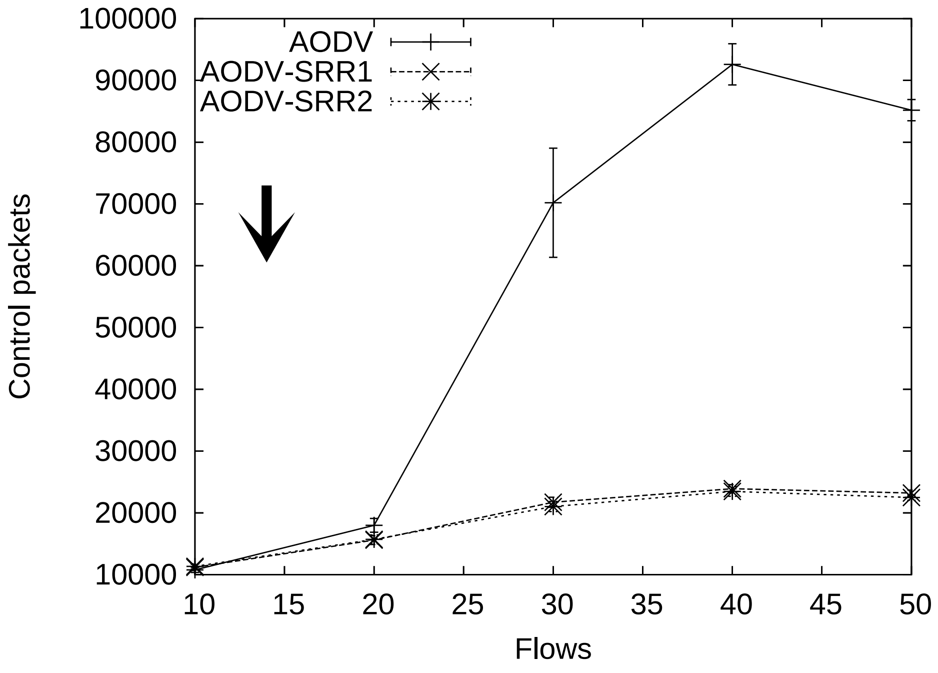
4.5.2 Results and Analysis

Varied CBR Flows

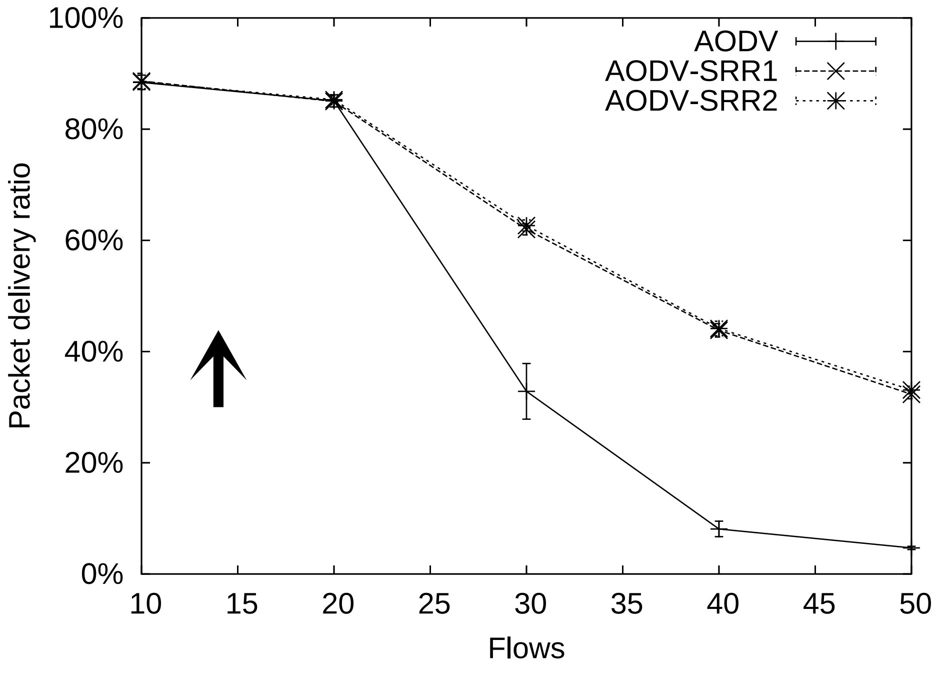
Figures 4.4(a), 4.4(b), and 4.4(c) show the number of control packets per flow, the PDR, and the end-to-end delay, respectively, when the number of flows varies from 10 to 50. When the number of flows is less than 20, AODV-SRR1 and AODV-SRR2 do not show much improvement over AODV. However, when the number of flows increases, the improvement becomes significant. For example, when there are 40 flows, compared to AODV, AODV-SRR1 and AODV-SRR2 reduce the number of control packets by 74% (Figure 4.4(a)), improve the PDR from 8% to 44% (Figure 4.4(b)), and reduce the end-to-end delay by 78% and 84% (Figure 4.4(c)), respectively. SRR significantly improves the performance of AODV in all aspects, not trading off one performance aspect for another. SRR2 has a slightly better performance than SRR1.

The improvement verifies the motivation discussed in Section 4.2. RREP packets in on-demand routing protocols are important, and a routing protocol should take special care to prevent them from being lost. SRR may not succeed in every attempt, but every RREP it salvages saves a large number of RREQ transmissions. SRR reduces the end-to-end delay as well because packets spend less buffer time waiting for route replies. As a result, nodes discard fewer packets from their buffers due to overflow or timeout, so the packet delivery ratio is also improved.

When the number of flows increases, the network carries more traffic, and more RREP packets

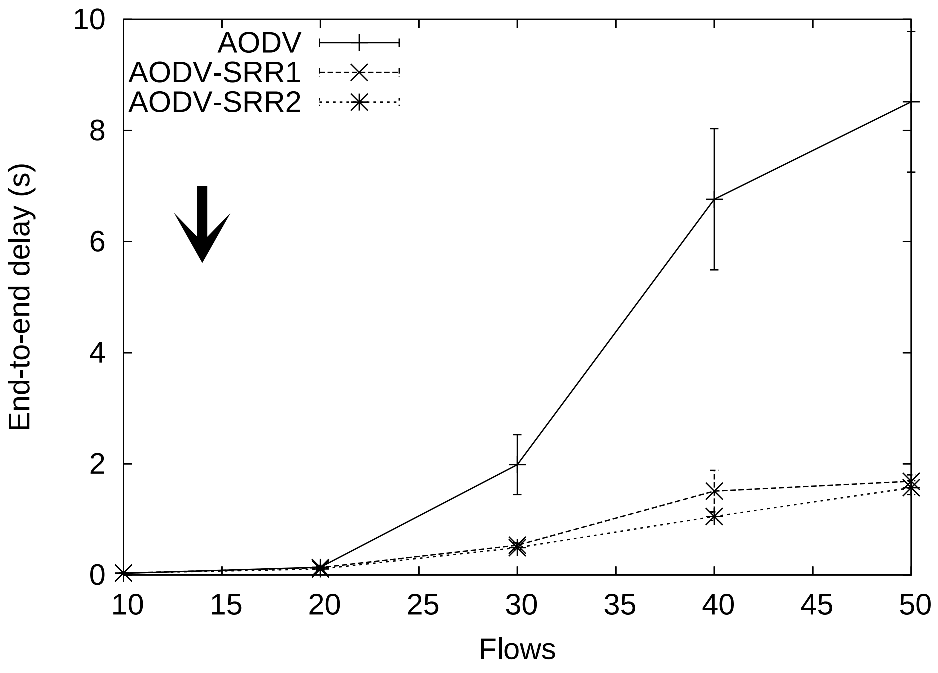


(a) Control packets.

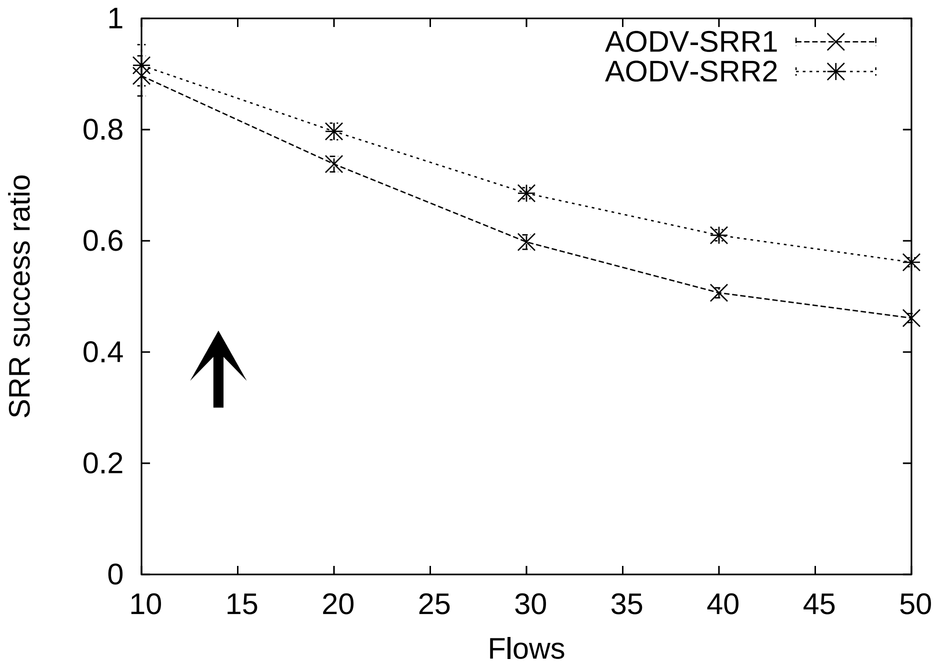


(b) Delivery ratio.

Figure 4.4: Performance when number of flows varies. Network area $1400 \times 1400 m^2$, 100 nodes, speed [0.1-20]m/s, and pause time $30s$.

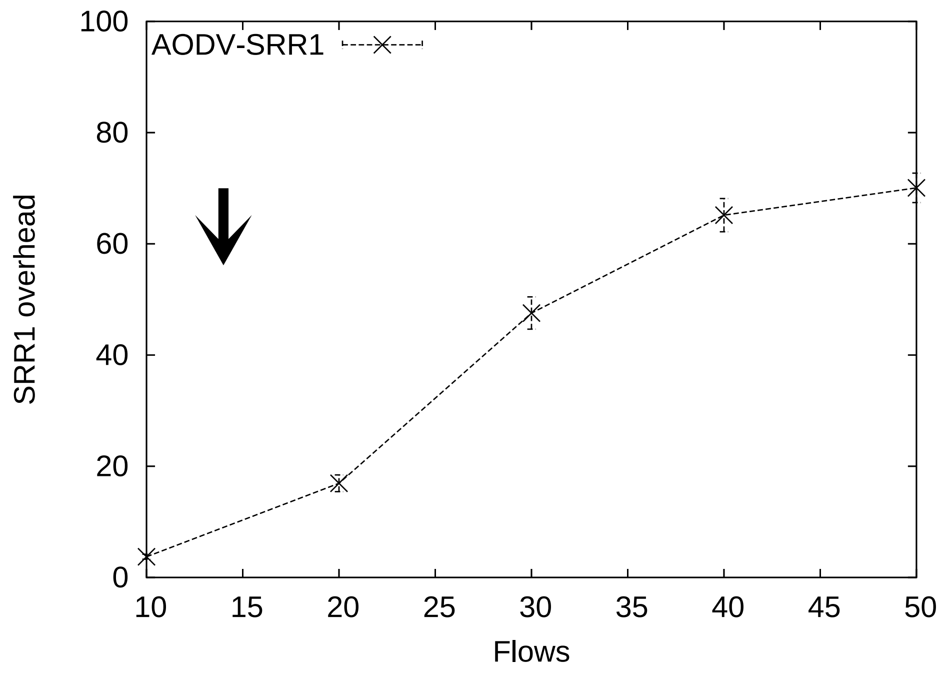


(c) End-to-end delay.



(d) SRR success ratio.

Figure 4.4: Performance when number of flows varies. Network area $1400 \times 1400 m^2$, 100 nodes, speed [0.1-20]m/s, and pause time $30s$ (continued).



(e) SRR1 overhead.

Figure 4.4: Performance when number of flows varies. Network area 1400×1400m^2, 100 nodes, speed [0.1-20]m/s, and pause time 30s (continued).

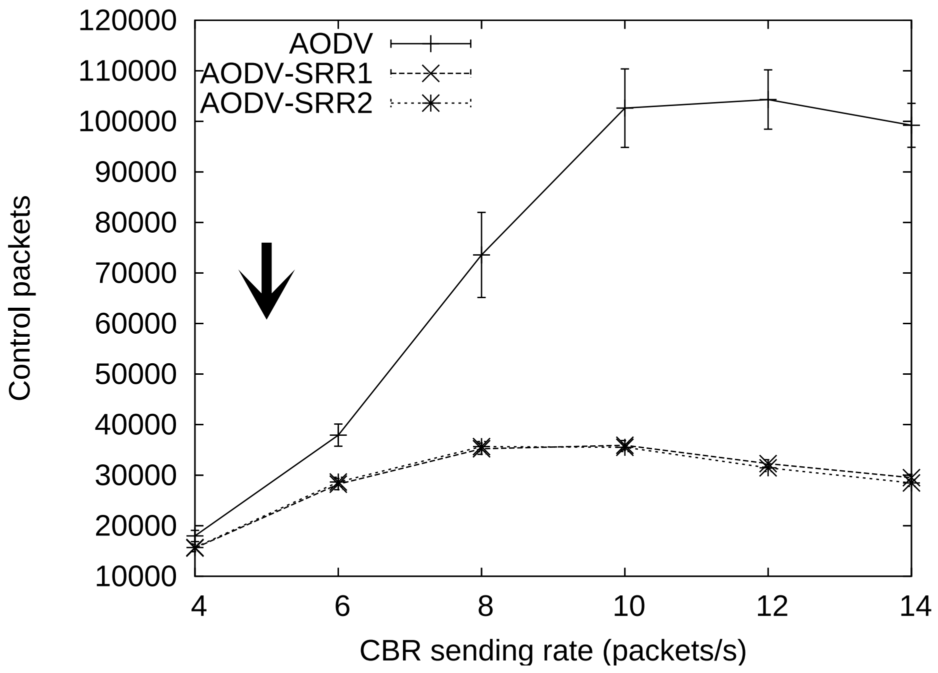
fail due to link breakages or channel congestion. Therefore, SRR salvages more RREP packets and improves the performance of AODV more significantly when the number of flows is higher. AODV-SRR2 performs slightly better than AODV-SRR1. One advantage of SRR2 is that it does not use extra control messages. But compared to the overall control overhead caused by the routing protocol, the control overhead saved by SRR2 may be small. Therefore, the performances of AODV-SRR1 and AODV-SRR2 are close to each other. However, compared to SRR1, in addition to no extra control messages, SRR2 requires fewer changes to the routing protocol and thus simplifies the protocol design. In SRR2, we do not need to design the formats of SREQ and SREP messages and implement their respective message handlers.

Figure 4.4(d) shows the success ratios of AODV-SRR1 and AODV-SRR2, and Figure 4.4(e) shows the number of SRR control packets (including SREQ, SREP, and RUPD) per flow generated by AODV-SRR1. We observe that around 50% to 90% salvage attempts succeed. AODV-SRR2 has a higher success ratio than AODV-SRR1 because it does not incur extra control packets. The success ratios decrease with the number of flows because when the network becomes more congested, even after being salvaged, RREP packets are likely to become undeliverable again. SRR salvages a

RREP at most once (Section 4.3.3). When the number of flow increases, AODV-SRR1 generates more SRR control packets, whereas AODV-SRR2 does not generate any extra control packets at all. Therefore, SRR2 is a noticeable improvement over SRR1.

Varied CBR Load

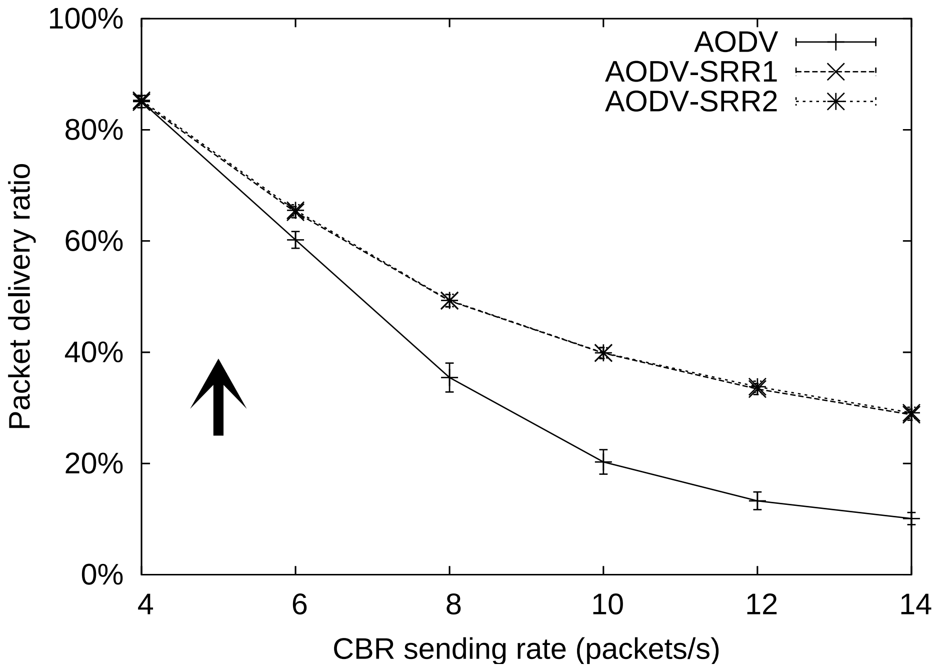
Figures 4.5(a), 4.5(b), and 4.5(c) show the number of control packets per flow, the PDR, and the end-to-end delay, respectively, when the CBR sending rate varies from 4 $packets/s$ to 14 $packets/s$. Like previous simulations, AODV-SRR1 and AODV-SRR2 have significantly better results than AODV for all three performance metrics. For example, when data packet sending rate is 10 packets per second, compared to AODV, AODV-SRR1 and AODV-SRR2 save the control overhead by more than 60%, improve the PDR by 97%, and reduce the end-to-end delay by about 45%.



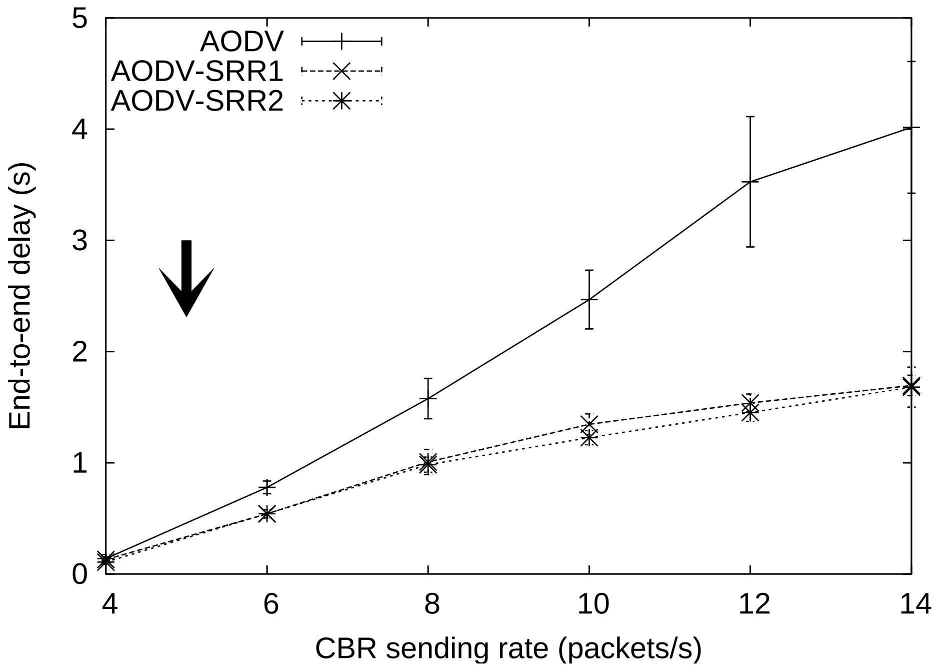
(a) Control packets.

Figure 4.5: Performance when CBR sending rate changes. Network area $1400 \times 1400 m^2$, 100 nodes, 20 flows, maximum speed $20 m/s$, and pause time $30 s$.

When the CBR sources inject more data packets to the network, the network becomes more congested, and more packets fail. Thus, SRR is able to salvage more RREP packets and improve the performance of AODV more significantly.



(b) Delivery ratio.

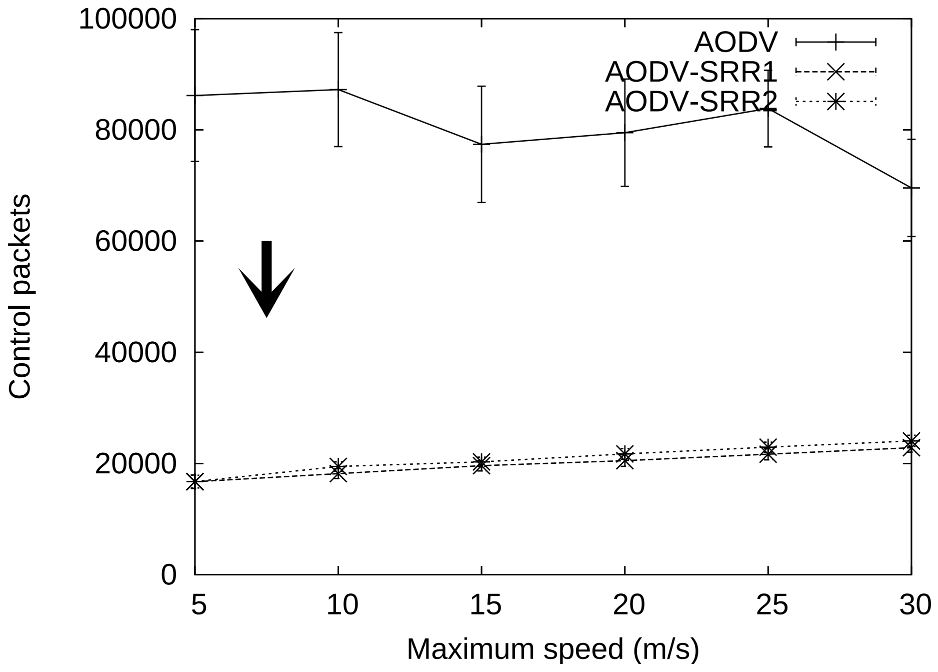


(c) End-to-end delay.

Figure 4.5: Performance when CBR sending rate changes. Network area $1400 \times 1400 m^2$, 100 nodes, 20 flows, maximum speed $20 m/s$, and pause time $30 s$ (continued).

Varied Maximum Node Speed

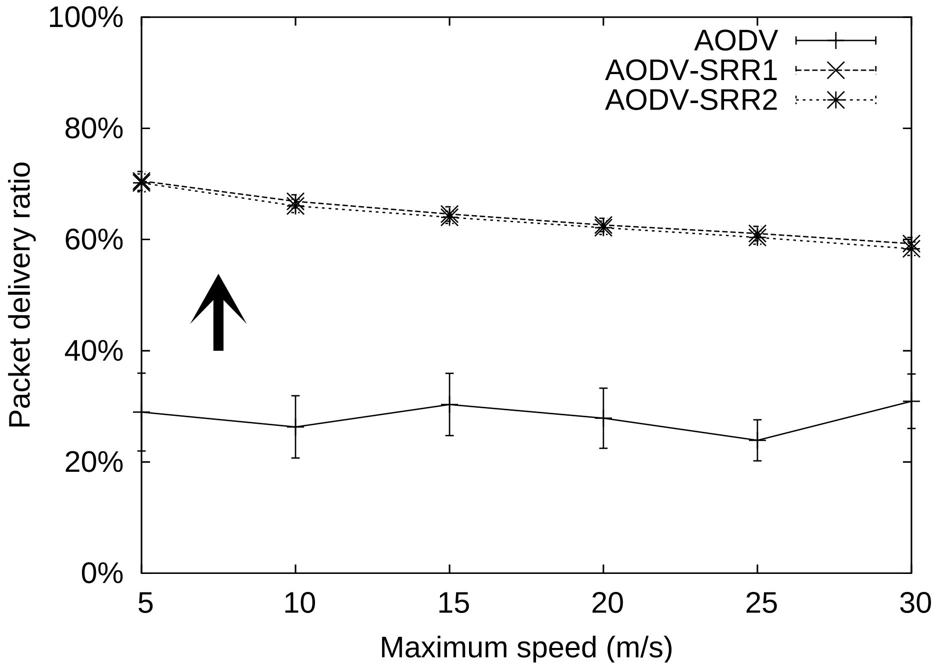
Figures 4.6(a), 4.6(b), and 4.6(c) show the number of control packets per flow, the PDR, and the end-to-end delay, respectively, when the maximum node speed varies from $5m/s$ to $30m/s$. The number of flows is 30 and the pause time is $0s$. From the results, we observe that AODV-SRR1 and AODV-SRR2 perform significantly better than AODV by all three performance metrics. For example, when maximum node speed is $20m/s$, AODV-SRR1 and AODV-SRR2 approximately save the number of control packets by 73%, improve the PDR from 28% to 63%, and reduce the end-to-end delay by 82%.



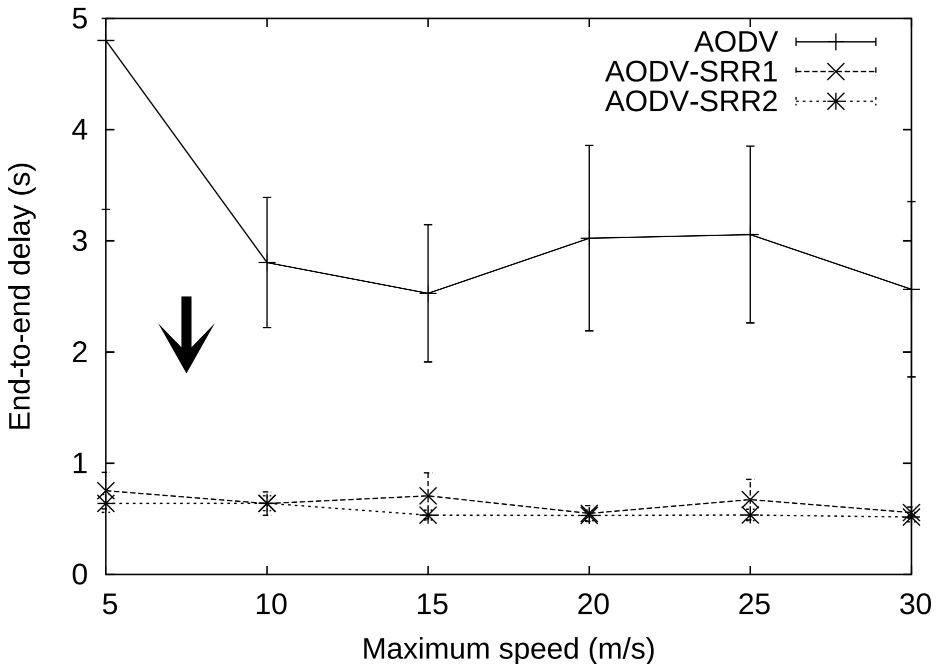
(a) Control packets.

Figure 4.6: Performance when maximum node speed changes. Network area $1400 \times 1400m^2$, 100 nodes, 30 flows, and pause time $0s$.

When nodes in the network move faster, links break more often and more packets are not deliverable. Thus, SRR salvages more RREP packets and improves the performance significantly. However, since the network topology changes more dramatically as the node speed increases, the routes established by the salvaged RREPs may break soon. Thus, the performance improvement is more significant at lower speeds.



(b) Delivery ratio.

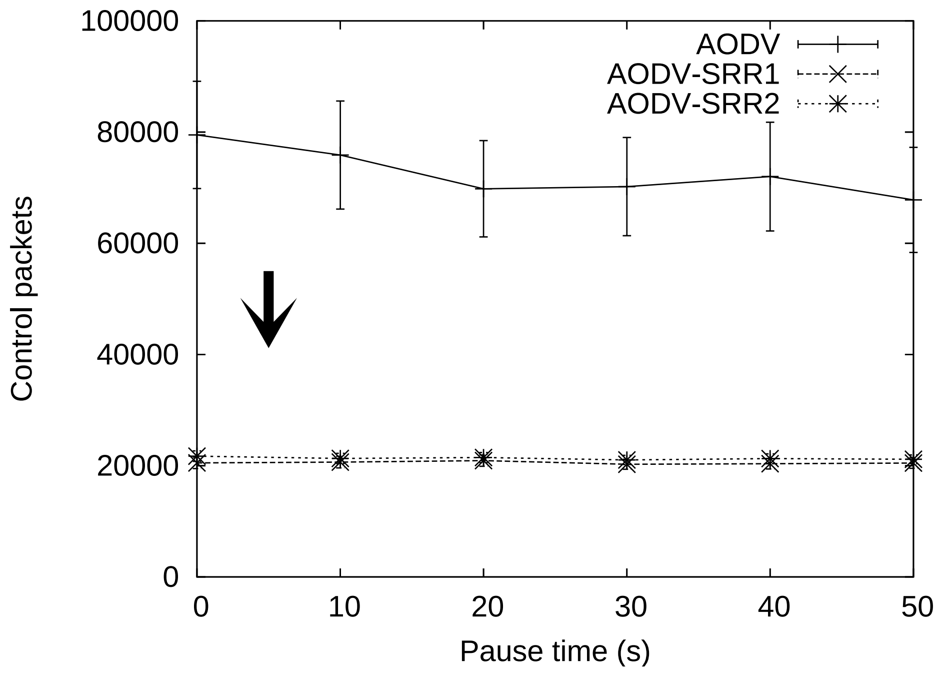


(c) End-to-end delay.

Figure 4.6: Performance when maximum node speed changes. Network area $1400 \times 1400 m^2$, 100 nodes, 30 flows, and pause time $0s$ (continued).

Varied Pause Time

Figures 4.7(a), 4.7(b), and 4.7(c) show the number of control packets per flow, the PDR, and the end-to-end delay, respectively, when the node pause time varies from $0s$ to $50s$. The number of flows is 30 and the maximum node speed is $20m/s$. We observe that AODV-SRR1 and AODV-SRR2 perform considerably better than AODV by all three performance metrics. For example, when the node pause time is $20s$, AODV-SRR1 and AODV-SRR2 reduce the number of control packets by approximately 70%, improve the PDR from 30% to 60%, and reduce the end-to-end delay by 60%.

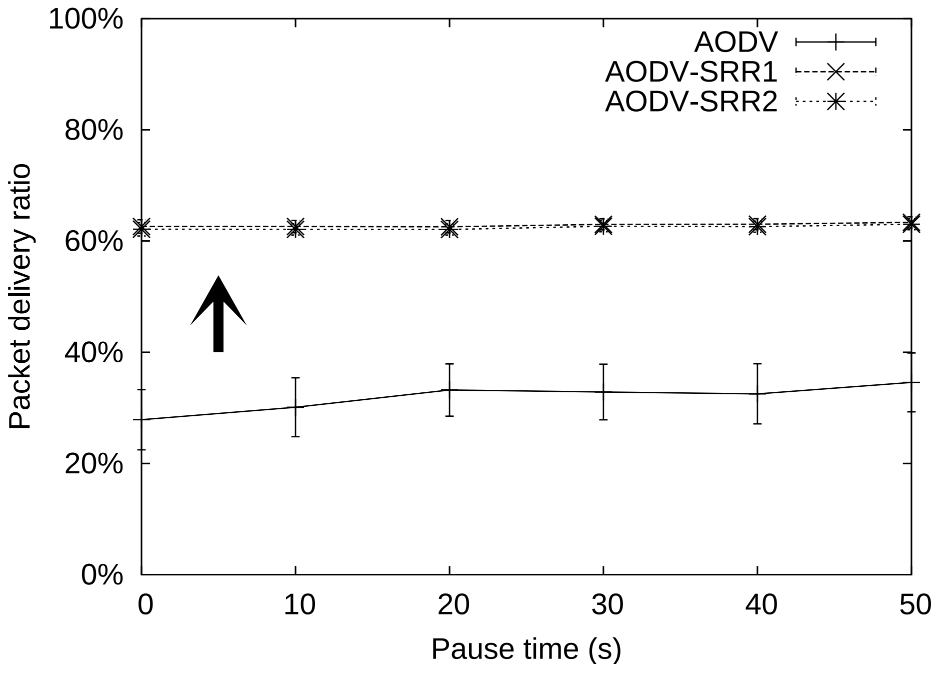


(a) Control packets.

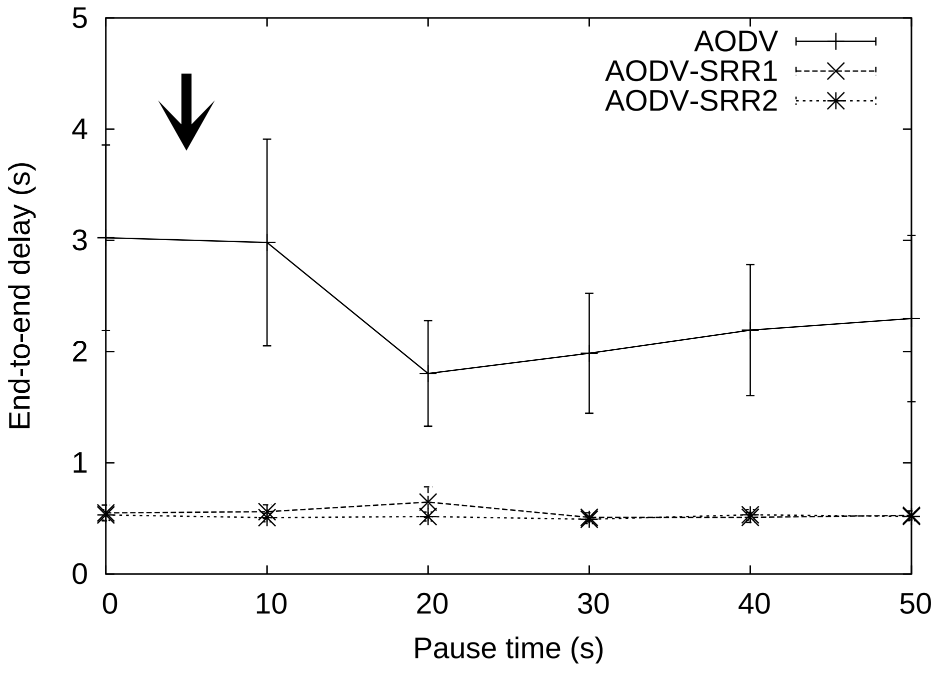
Figure 4.7: Performance when pause time changes. Network area $1400 \times 1400m^2$, 100 nodes, 30 flows, and maximum speed $20m/s$.

Varied Network Size

In this set of simulations, we study the performance of AODV-SRR1, AODV-SRR2, and AODV when the size of networks increases from 100 nodes to 300 nodes. Figures 4.8(a), 4.8(b), and 4.8(c) show the number of control packets per flow, the PDR, and the end-to-end delay, respectively. The performance of AODV degrades rapidly with increasing network size. For example, in networks with 200 nodes and 40 flows, AODV delivers only 7% of data packets. AODV-SRR1 and AODV-SRR2 improve the performance significantly. For example, when the network size is 200, AODV-



(b) Delivery ratio.



(c) End-to-end delay.

Figure 4.7: Performance when pause time changes. Network area $1400 \times 1400 m^2$, 100 nodes, 30 flows, and maximum speed $20 m/s$ (continued).

SRR improves the delivery ratio from 7% to 43%, reduces the number of control packets by 69%, and reduces the end-to-end delay by 76%.

As the network size grows, the route between a source and a destination becomes longer. A RREP packet has to travel farther before reaching the source node. It is more likely that the RREP cannot reach its intended next hop. Therefore, AODV-SRR1 and AODV-SRR2 have more chances to salvage undeliverable RREPs and play a more important role in improving routing performance.

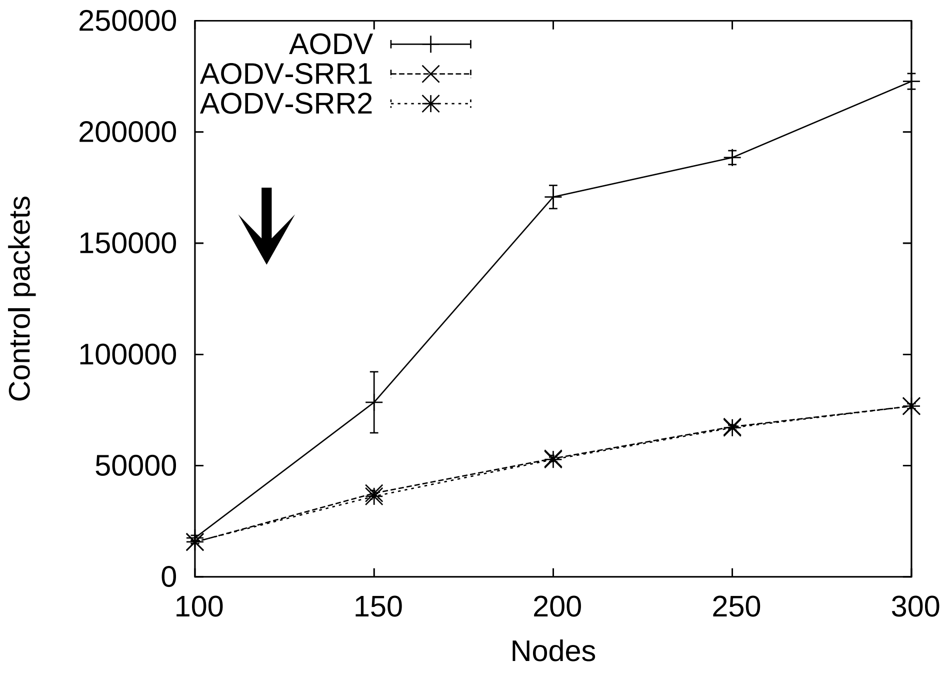
4.6 Summary

Route reply (RREP) messages are important in on-demand routing protocols for ad hoc networks. The loss of RREPs seriously impairs routing performance because the cost of a RREP is very high. Typically, an on-demand routing protocol obtains a RREP after flooding the entire or a part of the network with route request (RREQ) messages. This process is expensive and time-consuming — a route discovery requires tens, maybe hundreds of RREQ transmissions. If the RREP is lost, a large amount of route discovery effort will be wasted. Furthermore, the source node may have to initiate another round of route discovery to establish a route to the destination.

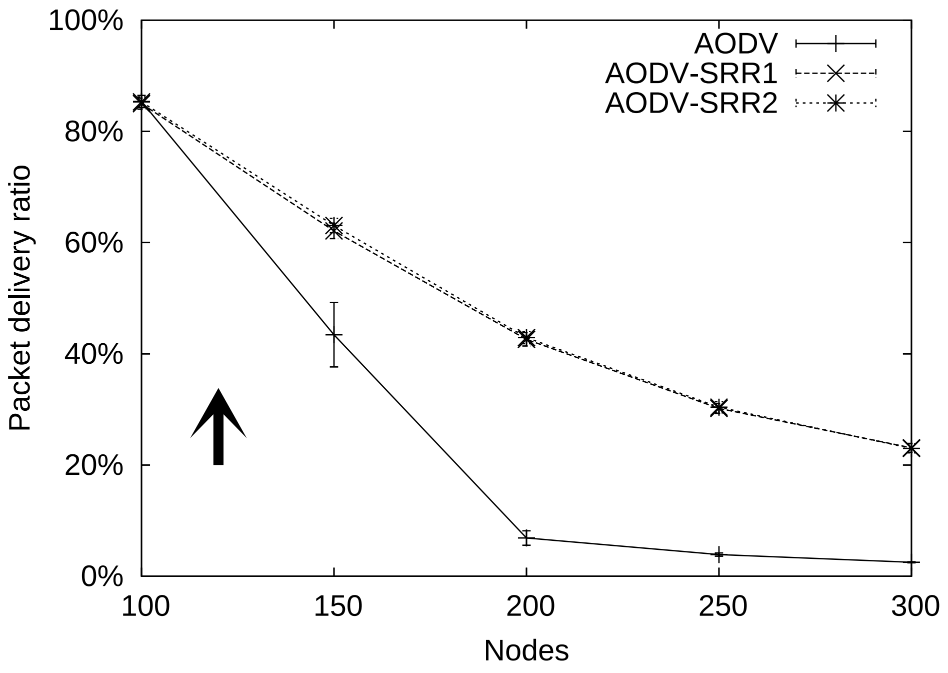
We propose the idea of salvaging route reply (SRR), which attempts to salvage an undeliverable RREP in two schemes. In SRR1, the salvor runs a one-hop SRR route discovery to find an alternative path to the source. The SRR route discovery succeeds most of the time because neighboring nodes recently propagated the RREQ message originated from the source and learned a reverse path to the source. In SRR2, intermediate nodes utilize duplicate RREQ packets and maintain a backup path to the source. When an intermediate node cannot relay a RREP message using the original path to the source, it directly switches to the backup path. SRR2 is an improvement to SRR1 because SRR2 requires no extra control message when salvaging an undeliverable RREP.

We present the implementation of both SRR schemes in AODV. We prove that in the implementation, routes are loop-free after a salvage. We conduct extensive simulations to evaluate the performance of two SRR schemes in conjunction with AODV. The results show that SRR significantly improves routing performance in a wide range of system parameter values and in all critical metrics, including packet delivery ratio, control overhead, and end-to-end delay.

This research is the first to our knowledge that focuses on the loss of route reply messages and proposes salvaging route reply messages to enhance routing performance in MANETs. Our

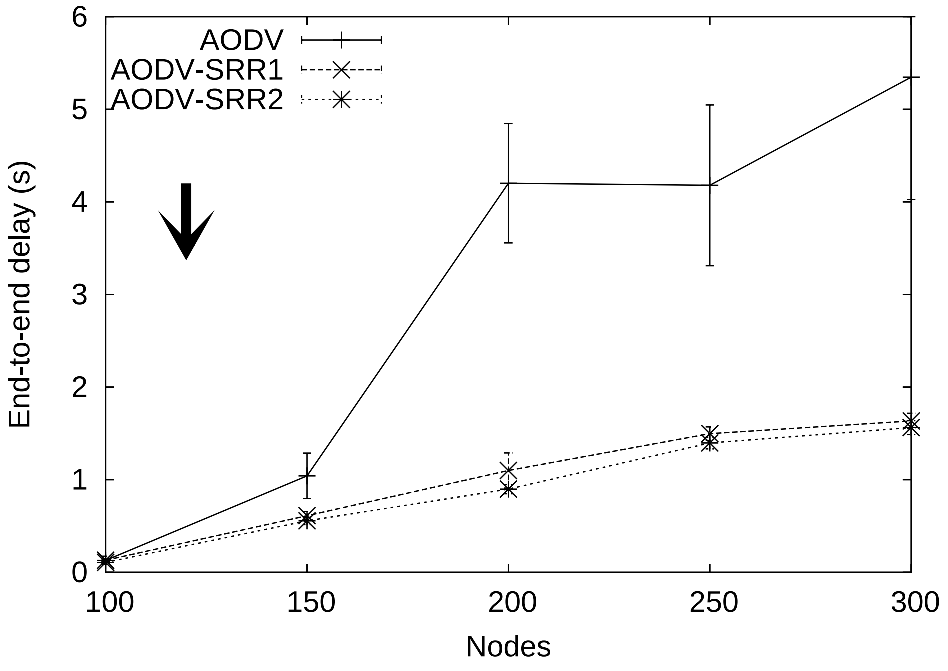


(a) Control packets.



(b) Delivery ratio.

Figure 4.8: Performance when network size increases. Constant node density, flows 20% of nodes number, maximum speed $20m/s$, and pause time $30s$.



(c) End-to-end delay.

Figure 4.8: Performance when network size increases. Constant node density, flows 20% of nodes number, maximum speed $20m/s$, and pause time $30s$ (continued).

approach is practical and applicable to all on-demand routing protocols and does not conflict with existing optimization techniques reviewed in Chapter 2.

Chapter 5

Carpooling in Mobile Ad Hoc Networks

5.1 Introduction

Routes in MANETs may break often due to factors such as node mobility and channel congestion. Upon each route breakage, the source node may have to conduct a new route discovery. Therefore, the dominant control overhead of on-demand routing protocols is RREQ packets. Usually, at any time, a considerable number of RREQ packets are being propagated in the network. This observation leads to our intuition for this work: When a node has to find a route to a destination, instead of immediately injecting a new RREQ message into the network, the node tries to discover a route by attaching its request to in-transit RREQ packets that it relays for other nodes. This approach has potential to improve performance by reducing the overhead, congestion, and power consumption at nodes.

We term this approach *multiple-target route discovery* (MTRD). MTRD is analogous to *carpooling* in real life. Carpooling is a more efficient transportation option than driving alone. Both carpooling and MTRD have the following two advantages:

(i) *Reducing traffic*. Carpooling reduces traffic on roads, and MTRD reduces traffic on communication channels. MTRD reduces the number of message floodings, which typically are expensive and time-consuming.

(ii) *Reducing cost*. Carpooling saves gas and parking, and MTRD reduces overhead at multiple layers because it aggregates route requests. First, MTRD reduces MAC/PHY overheads, such as MAC header and PLCP (physical layer convergence protocol) preamble/header [8], because there are fewer frames. Kim et al. [62] show that aggregating small-size frames into a large frame can significantly improve the throughput of IEEE 802.11. Second, MTRD reduces routing overhead because multiple requests can share common information. For example, multiple requests can share a common identifier, which typically uses the combination of a source address and a broadcast id. Moreover, if a route discovery scheme records the traversed path (like DSR [58]), multiple requests share the same path.

Therefore, as in real life, we should encourage and exploit carpooling in MANETs. We can use carpooling for both data and control packets, and we can use carpooling at both network and MAC/PHY layers. As a start in this direction, we study the aggregation of route requests. The contributions of this chapter are as follows:

(i) We exploit the idea of carpooling in MANETs to optimize on-demand routing. Particularly, we focus on the fact that usually a large number of RREQ packets are in transit in a MANET at any time. They provide opportunities of discovering routes for intermediate nodes that relay the RREQs. A node can attach its route request to a RREQ packet that it is relaying rather than initiating a new RREQ message.

(ii) We propose a new route discovery scheme called multiple-target route discovery (MTRD) for on-demand routing protocols. In MTRD, a RREQ message may have multiple target fields: one is the destination searched by the initiator of the RREQ, and others are attached by nodes that have relayed the RREQ.

(iii) We examine the performance of MTRD in conjunction with AODV [86] through an extensive simulation study. The results show that MTRD improves routing performance significantly when the network load is medium to high.

Routing protocols for MANETs occasionally use piggybacking. However, they use piggybacking for different purposes from MTRD. Existing piggybacking operations mainly focus on routing functionality, whereas we use carpooling for the purpose of improving the efficiency and economy of routing in ad-hoc networks. In the current Internet Draft of DSR [59], piggybacking may be used in three scenarios. First, when unidirectional links exist, upon receiving a RREQ, the destination piggybacks a RREP on a new RREQ targeted at the source. Second, upon receiving a route error message, a source may piggyback it on the following RREQ message that is in response to the route breakage. Third, a source may piggyback small data packets, such as TCP SYN [88], on a RREQ message to reduce the delay of these packets.

The rest of this chapter is organized as follows. The next section discusses motivations of MTRD. Section 5.3 describes the MTRD protocol. Section 5.4 presents simulation setup and results. We summarize this chapter in Section 5.5.

5.2 Motivations and Challenges

On-demand routing protocols typically establish a route through a route request/reply cycle. A source node initiates a route discovery process by broadcasting a RREQ message. Intermediate nodes relay the RREQ at most once. Upon receiving the RREQ, the destination sends a route reply (RREP) message to the source, which subsequently uses the discovered route to send data packets. We call this route discovery process a *regular route discovery*. In this section, we first discuss the motivations of MTRD and then discuss the challenges in designing MTRD.

5.2.1 The Motivations of MTRD

Due to the dynamic nature of MANETs, routes may break and need to be re-discovered often. Therefore, at any instance of time, a considerable number of RREQ packets are likely to be in transit in the network.

We conducted a set of simulations to study the number of RREQ packets in transit. The network had 100 nodes and the traffic load varied from 10 flows to 50 flows. Other setups were the same as described in Section 5.4. Figure 5.1 shows the average number of RREQ packets that a node transmits per second.

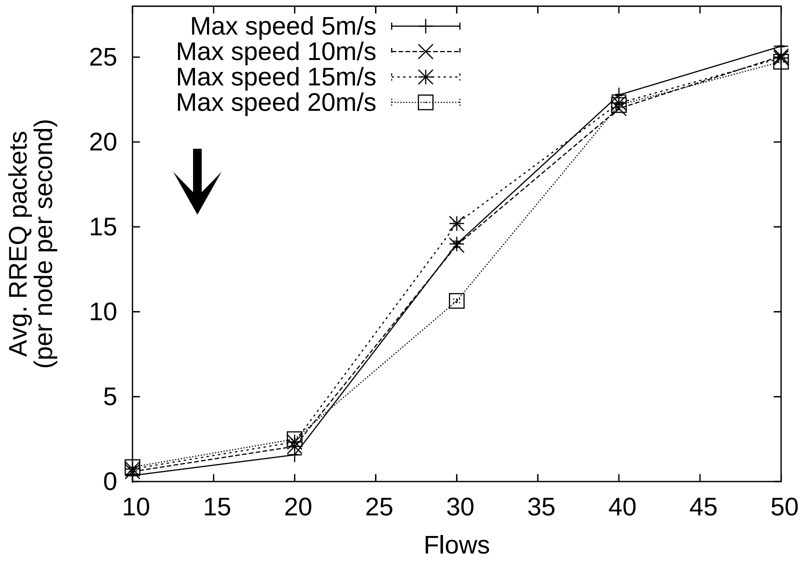


Figure 5.1: RREQ packets transmitted per node per second. 100 nodes, $1400 \times 1400 m^2$, four 512-byte packets per second CBR, min node speed $0.1 m/s$, max node speed 5-20m/s, and pause time $30s$.

We observe that a node in general relays a large number of RREQ packets for ongoing route discoveries. For example, when the number of flows is 30, on the average each node transmits about 10 to 15 RREQ packets per second. This significant number of request packets motivated us to consider utilizing in-transit RREQ packets for new route discoveries. In the simulations, RREQ packets accounted for more than 70% of the total number of control packets. This ratio indicates that RREQ packets are the major source of routing overhead, and reducing the number of RREQ packets will directly benefit routing performance.

When a node generates a route request, some RREQ packets that it relays for other sources might reach the destination to which the node needs a route. If the node attaches its request to these RREQs, the destination will receive the request and will send a reply to the node. In such a way, the source establishes a route to the destination, without running a regular route discovery process. If a node attaches its own route request to a RREQ packet that it is relaying, the RREQ packet will contain more than one target. Therefore, we term this new route discovery scheme multiple-target route discovery (MTRD).

Using MTRD has the following advantages. First, MTRD reduces the number of regular route discoveries, i.e., MTRD reduces control overhead. Second, lower control overhead reduces congestion and thus helps improve packet delivery ratio and reduce end-to-end delay. However, MTRD may not always succeed, so the time spent by MTRD may increase the end-to-end delay. We need to evaluate the end-to-end delay in the performance study.

5.2.2 Challenges in Designing MTRD

MTRD seems to be an effective idea, however, we need to consider a number of challenges carefully before MTRD can perform satisfactorily. The following are the challenges in designing MTRD:

(i) Selecting RREQs to attach. When using MTRD, it is not appropriate for a source node to attach its request to every RREQ packet that it relays, because some RREQs have less chance to reach the target. Otherwise, MTRD may unnecessarily increase the size of RREQ packets, which would consume more bandwidth and processing. We need to define certain conditions when selecting RREQs to attach. One criterion is TTL (time to live). For example, a RREQ with a TTL of 2 is not likely to reach a target with an estimated distance of 4 hops, and thus we should not use it for attaching.

(ii) Controlling the size of RREQ packets. This consideration is to reduce the network bandwidth and the processing overhead. We need to control the size of RREQ packets in two aspects. First, for an attached request, the extra information (like the destination's address and sequence number) added to a RREQ should be as small as possible. Second, the number of targets attached to one RREQ should be limited.

(iii) Detecting a MTRD failure. MTRD may fail in two cases. First, a source node cannot find an appropriate RREQ to attach its request within some time. Second, a source node attaches its request to a number of RREQs, but none of them reaches the destination. We need to detect a MTRD failure quickly to reduce route discovery latency. If we use timers to detect the failure, the timeout settings may affect performance considerably.

(iv) Use MTRD or regular route discovery? If most nodes use MTRD for route discoveries, there may not be enough RREQs for attaching. So we should design the system such that nodes conduct enough regular route discoveries to ensure that RREQs are present for MTRD.

(v) The format of a RREQ message. To attach additional targets, we need to modify the format of a RREQ message. However, the modifications should be as small as possible. Particularly, when a RREQ message has no attached target, it should comply with the original format. Designing RREQ message in this manner facilitates the enabling and the disabling of MTRD. When disabled, the routing protocol operates the same as its original version.

5.3 Multiple-Target Route Discovery

Among many existing on-demand routing protocols, AODV (ad hoc on-demand distance vector) [86] is one of the most well-known and most studied protocols. We mainly focus on the design of MTRD in AODV, which we call AODV-MTRD. In this section, we first explain the RREQ message of AODV-MTRD, then present the operation of AODV-MTRD, and finally give an example of how AODV-MTRD works.

In the discussion, we call the node that originates a RREQ message the $initiator$, and call the node that demands a route to a target the $source$. By a route request, we mean the need for a route, instead of an actual message or packet (unless noted otherwise).

5.3.1 A RREQ Message of AODV-MTRD

RREQ messages of existing on-demand routing protocols such as AODV and DSR contain one target field. For the purpose of carrying multiple targets, we add two new fields to a RREQ message: att_target_count and att_target_array.

The *att_target_count* field records the number of attached targets. The storage of this field utilizes unused bits in a RREQ message (a RREQ message of AODV has 11 unused bits in its first quartet [82]), and thus it does not increase the message size. Generally, 2 or 3 bits are enough to store att_target_count. We use a system parameter called MAX_ATT_TARGETS to limit the number of targets that can be attached to a RREQ message.

The *att_target_array* field is a dynamic array that stores attached targets. The type of array is a structure composed of three variables: $target\_address$, $target\_seq$, and hop_count. The att_target_array field is placed at the end of a RREQ message and contains att_target_count targets.

Both MTRD and regular route discovery are used in AODV-MTRD. Typically, when a node has a route request, it tries MTRD first. If MTRD does not succeed, the node conducts a regular route discovery. Nodes in AODV maintain a request table to record route requests that they initiate. We define the following three status types to distinguish different statuses of a route request in the request table:

- *REQ_NON:* The request has neither been attached to an independent RREQ nor has a regular route discovery been conducted.
- *REQ_ATT:* The request has been attached (maybe more than once) to independent RREQs, but a regular route discovery has not been conducted yet.
- *REQ_REG:* The source is conducting a regular route discovery.

5.3.2 The Operation of AODV-MTRD

MTRD consists of an *attaching phase*, during which a source tries to attach its request to RREQs it relays, and a *waiting phase*, during which the source waits for a route reply. These two phases use timers called $CheckAttached$ and $CheckReplied$, respectively. In the attaching phase, we randomize the MTRD attempt time by setting the $CheckAttached$ timer uniformly between a lower bound (ATT_WAIT_LOW) and an upper bound (ATT_WAIT_UP). The purpose is to prevent all sources from waiting for RREQs to attach whereas no one is generating RREQ traffic. Next, we discuss in detail how a node attaches its request to a RREQ packet that it is relaying.

Attaching a Route Request

Figure 5.2 shows the flowchart of how a node attaches its request to a RREQ packet. The following conditions must be satisfied before the request can be attached:

(i) The request has not been conducted as a regular route discovery, i.e., the status of request is REQ_NON or REQ_ATT.

(ii) The TTL of the RREQ packet is greater than or equal to the estimated TTL of the node's request. The TTL of a route discovery is estimated by the TTL of a previous known route and by the expanding ring search scheme.

(iii) The number of existing targets attached in the RREQ (i.e., att_target_count) is less than MAX_ATT_TARGETS, which was set to 3 in our simulations.

(iv) If the target of an attached request in the RREQ is the same as the destination of this node's request, the request should not be attached, because the node may learn a route if the destination receives the RREQ and sends a route reply.

(v) If the node has more than one destination to attach (e.g., the node is the source for multiple flows), the node first selects a destination that has not been attached. Then the node selects a destination that has been attached the fewest times.

If the above conditions are satisfied, the node attaches its request to the RREQ. The att_target_count of the RREQ is increased by one, and the destination is appended to the att_target_array of the RREQ. The node also sets the status of the request in its request table to REQ_ATT if it is REQ_NON. Note that a request may be attached to multiple RREQ packets. The attaching phase ends when the $CheckAttached$ timer expires. When the attaching phase ends, if the request has been attached, MTRD goes to the waiting phase. If no route is discovered when the waiting phase ends or the request has not been attached (i.e., its status is REQ_NON), a regular route discovery is conducted.

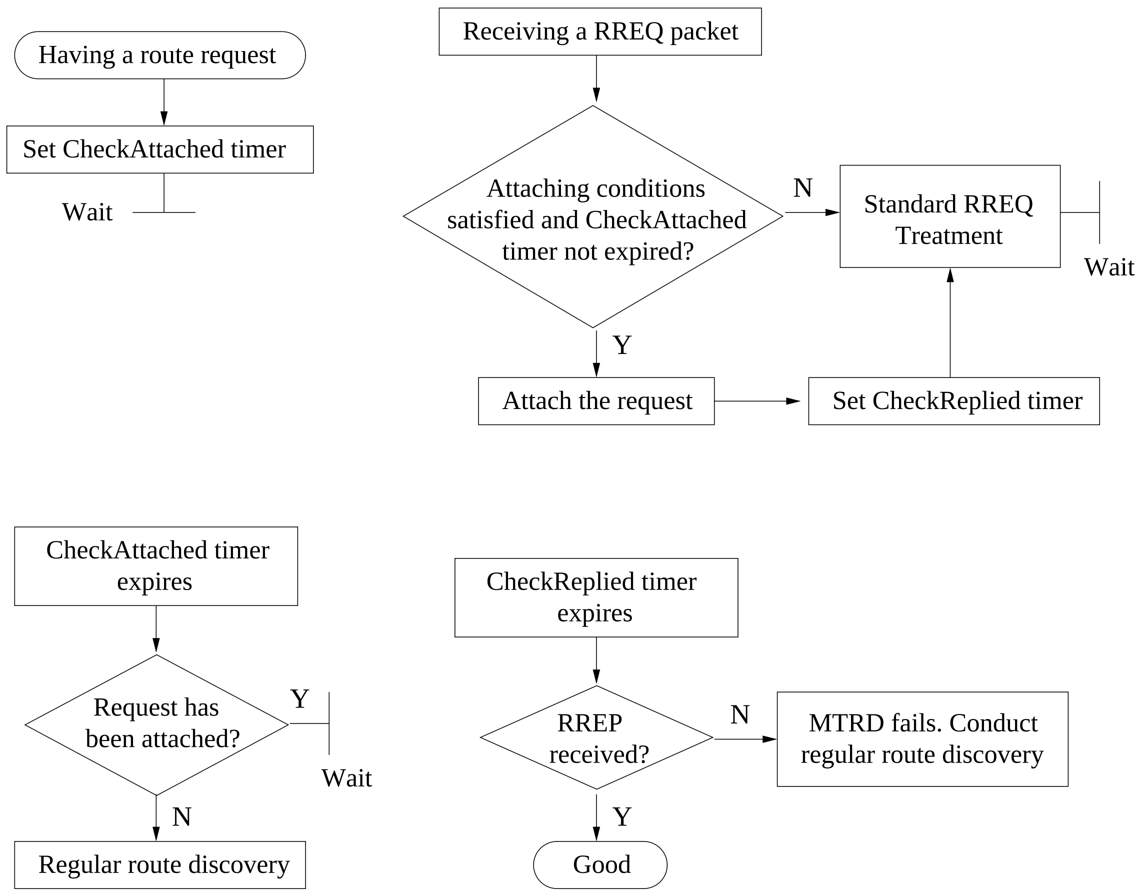


Figure 5.2: Attaching a route request.

Handling a Received RREQ Message

Figure 5.3 shows the flowchart of how a node handles a received RREQ message. The node first checks if its address matches one of the targets in the RREQ. This checking is done for all RREQ packets, including duplicate RREQ packets. The reason is that a duplicate RREQ packet may contain a new attached request. However, the same request attached to multiple RREQs is answered at most once, and duplicate RREQ packets are not relayed. Depending on whether the node is the target of an attached request, the handling of the RREQ has two cases:

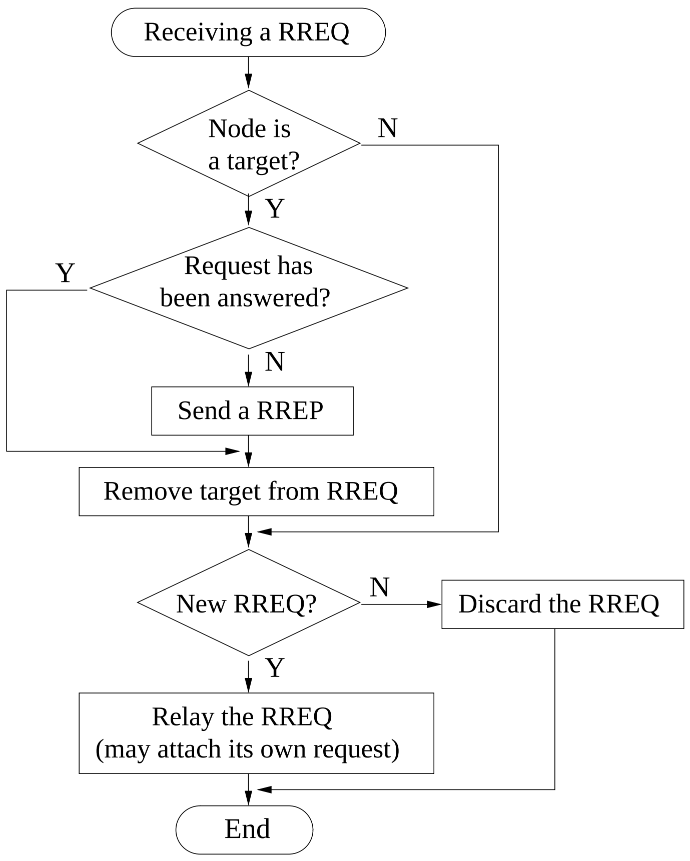


Figure 5.3: Handling a received RREQ message.

Case 1 (non-target): The RREQ is handled similarly as a RREQ is handled in AODV. A new RREQ packet is processed, whereas a duplicate RREQ packet is discarded. An intermediate node typically relays the RREQ, and the destination of the original RREQ sends a RREP to the initiator. When intermediate nodes relay a RREQ, AODV-MTRD differs from AODV in two ways. First, if the RREQ contains attached targets (i.e., $att\_target\_count > 0$), the hop_count for each target in the att_target_array is increased by one. Second, if the node has its own route request, it may attach the request to the RREQ, as described previously in Section 5.3.2.

Case 2 (target): If the request has not been answered, the node sends a route reply to the source, which we discuss in detail next. After sending the reply, the node removes the attached request from the RREQ and handles the RREQ as discussed in the Case 1.

Replying an Attached Route Request

When a source node attaches its request to a RREQ, it attaches the information for the destination, but its own information is not included to control the size of RREQ packets. We denote the reply message as RREP$_{att}$. To let a source know that a RREP$_{att}$ is for itself, we define a field called $total\_hop\_count$ in the RREP$_{att}$. The $total\_hop\_count$ is the distance (hops) from the source to the destination. The RREP$_{att}$ also has the same hop_count field as a regular RREP. When the RREP$_{att}$ reaches the source, the hop_count is equal to the $total\_hop\_count$ and thus the source knows the reply is for itself. This completes the route discovery process.

5.3.3 An Example of MTRD

We use an example in Figure 5.4 to show how AODV-MTRD works. Initiator A is discovering a route to destination E. Node B attaches its own route request for node G to the RREQ. When node G receives the RREQ relayed by node F, G finds that it is a target and sends a RREP$_{att}$ with the $total\_hop\_count$ set to 2. When node B receives the RREP$_{att}$, it knows the message is for itself because both the hop_count and the $total\_hop\_count$ are 2. Thus, node B successfully discovers a route to node G.

Figure 5.4 also shows changes to the RREQ message along the path $A \rightarrow B \rightarrow F \rightarrow G$, and changes to the RREP$_{att}$ message along the path $G \rightarrow F \rightarrow B$. Information such as broadcast id and sequence number is ignored.

5.4 Performance Evaluation

We conduct extensive simulations to evaluate the performance of AODV-MTRD and compare it with AODV. In this section, we first introduce the simulation setup and then present the results and analysis.

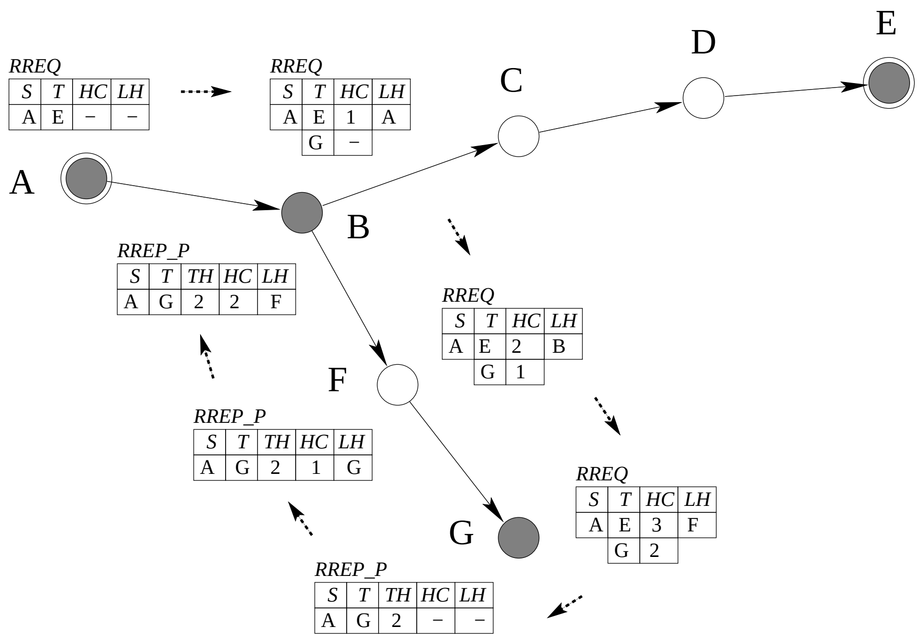


Figure 5.4: An example of how AODV-MTRD works. Source B attaches its request to RREQ from initiator A. $S-$ source, $T-$ target, $HC-$ hop count, $LH-$ last hop, and $TH-$ total hop count.

5.4.1 Simulation Setup

We conduct simulations using GloMoSim 2.03 [21]. The settings for radio model and MAC layer are the same as described in Section 3.4. The traffic is constant bit rate (CBR), and the mobility model is random waypoint [23]. We select CBR sources and destinations randomly. A flow does not change its source and destination during the lifetime of a simulation run. Each simulation run lasts for 1200 seconds. Each source waits for a random time from 60 to 100 seconds before sending data packets, and stopped sending data packets 20 seconds before the simulation ends. The results in the graphs are averaged over 20 simulation runs, each with a different seed. Error bars represent the 95% confidence interval for each data point. We add a "goodness arrow" to each graph showing which direction is better.

Unless specified otherwise, the following configurations for the traffic and the mobility are common: CBR sending rate four 512-byte packets per second, minimum node speed 0.1 m/s, maximum node speed 20 m/s, and pause time 30s. We set system parameters for MTRD as follows: MAX_ATT_TARGETS (maximum number of targets that can be attached to a RREQ packet) is 3; ATT_WAIT_LOW and ATT_WAIT_UP (lower and upper bounds for the MTRD attempt timer) are 200 ms and 600 ms, respectively.

We first study AODV-MTRD and AODV in networks containing 100 nodes in an area of 1400×1400 m^2. We group these simulations into three sets by varying three network parameters, namely, number of flows (Section 5.4.2), CBR sending rate (Section 5.5), and maximum node speed (Section 5.5), respectively. Table 5.1 summarizes the setups for the 100-node simulations.

Table 5.1: The setup summary of 100-node simulations.

Parameter	*Flows*	*CBR Rate* $(pkts/s)$	*Max Speed* (m/s)	*Pause Time* (s)
Flows	10 – 50	4	20	30
CBR Rate	20	4 – 12	20	30
Max Speed	25	4	5 – 25	0

We then studied AODV-MTRD and AODV in larger networks, where the number of nodes varied from 100 to 300 (Section 5.7). The number of flows was 25% of the total number of nodes, and the node density was kept constant by increasing network area accordingly. Table 5.2 gives the setup for this set of simulations.

Table 5.2: Network size, number of flows and area.

Size	*Flows*	*Area* (m^2)
100	25	1400×1400
150	38	1700×1700
200	50	2000×2000
250	63	2200×2200
300	75	2450×2450

We measure three widely used metrics: *packet delivery ratio (PDR)*, *control overhead*, and *end-to-end delay*. Moreover, we measure three additional metrics for the 100-node simulations with varied CBR flows: ratio of MTRD received to attached, byte overhead of RREQ packets, and route discovery delay. We define the ratio of MTRD received to attached as the ratio of the number of RREQ packets that are received by the targets of attached requests to the number of RREQ packets that sources attach their requests to. This ratio indicates the probability of an attached route request being received by the intended target.

5.4.2 Results and Analysis

Varied CBR Flows

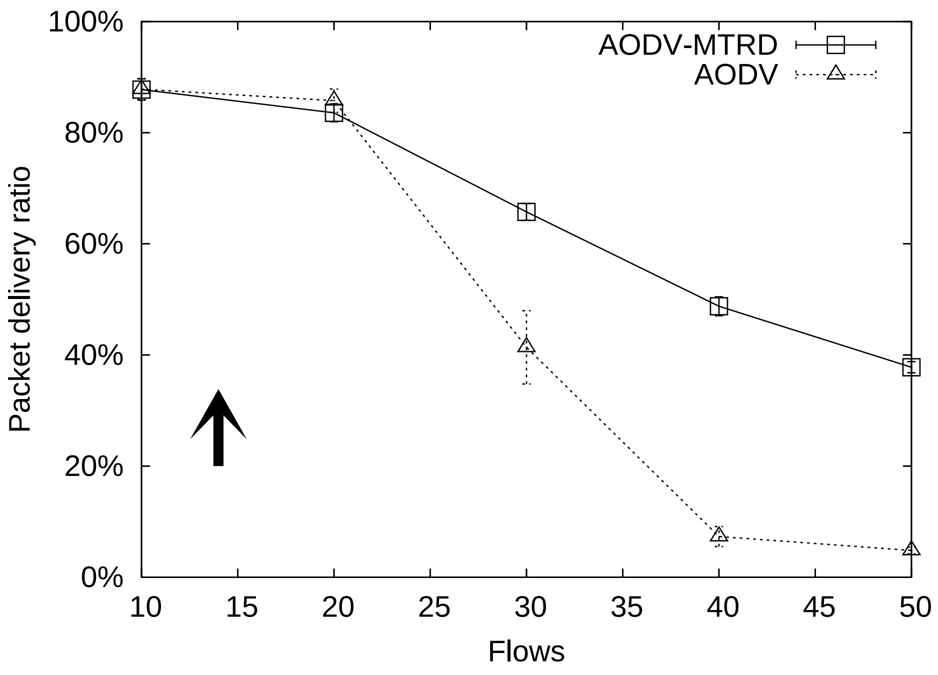
Figures 5.5(a), 5.5(b), and 5.5(c) show the results of PDR, control overhead per flow, and end-to-end delay, respectively, when the number of flows varies from 10 to 50. We observe that with increasing number of flows, MTRD improves the performance of AODV significantly. For example, when there are 30 flows, the PDR improves by 59% (Figure 5.5(a)), the control overhead is reduced by 71% (Figure 5.5(b)), and the end-to-end delay is slightly better (Figure 5.5(c)). The curve in Figure 5.5(b) goes down at the 50-flow point because the network is reaching its load capacity and cannot deliver many more packets than the 40-flow case.

These improvements verify our motivation: ongoing route discoveries can be utilized by intermediate nodes to discover new routes. MTRD considerably reduces the overhead of RREQ packets incurred by on-demand routing protocols. Fewer RREQ packets means less congestion, and thus fewer data packets are dropped due to collision and due to queuing time-out. AODV-MTRD has better or comparable end-to-end delay than AODV. MTRD reduces congestion, and data packets are likely to travel faster, but MTRD may have a slightly longer route discovery delay because it needs to wait for appropriate RREQ packets.

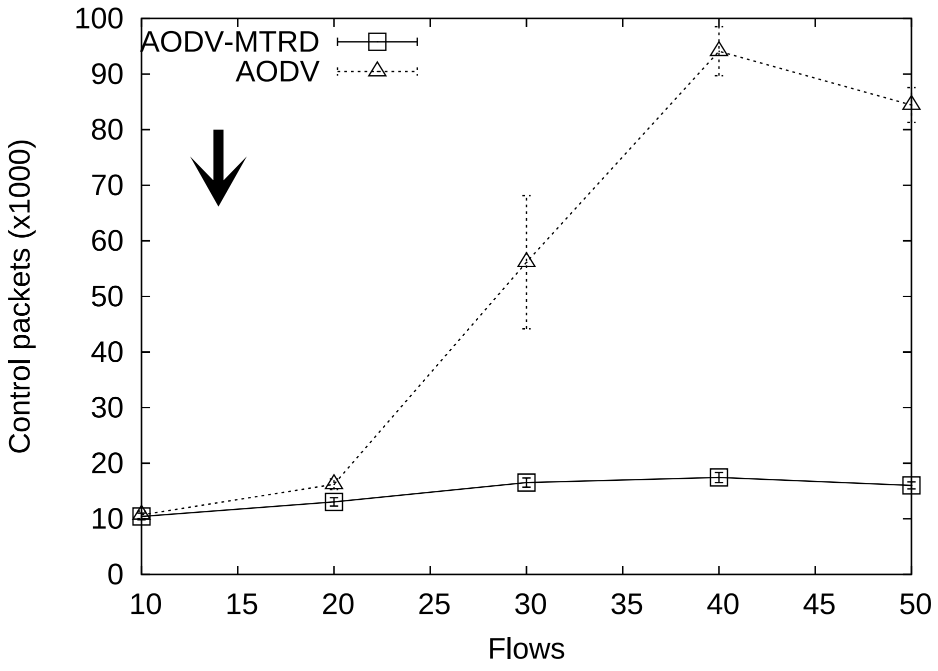
Twenty plus flows for a 100-node network might seem a bit heavy. However, these scenarios are not unrealistic. Increasing interest in MANETs will result in increasing traffic; in addition, network traffic is often bursty. It is important for routing protocols to perform well during "rush hours".

Perhaps a misleading opinion is that proactive routing may perform better than on-demand routing when network load becomes heavy. In fact, Broch et al. [23] discover that DSDV fails to converge when pause time (random waypoint mobility) is below 300 seconds, and its performance drops dramatically, whereas AODV and DSR perform much better. Johansson et al. [57] give similar results.

Figure 5.5(d) shows the ratio of MTRD received to attached. We observe that on the average the ratio is about 0.4. This ratio indicates that an attached route request reaches its target with a reasonable probability. Figure 5.5(e) shows the byte overhead of RREQ messages per flow in AODV-MTRD and in AODV. MTRD significantly reduces the byte overhead of RREQ messages. We did not count IP headers when measuring the byte overhead. The improvement would be more significant if IP headers were counted.

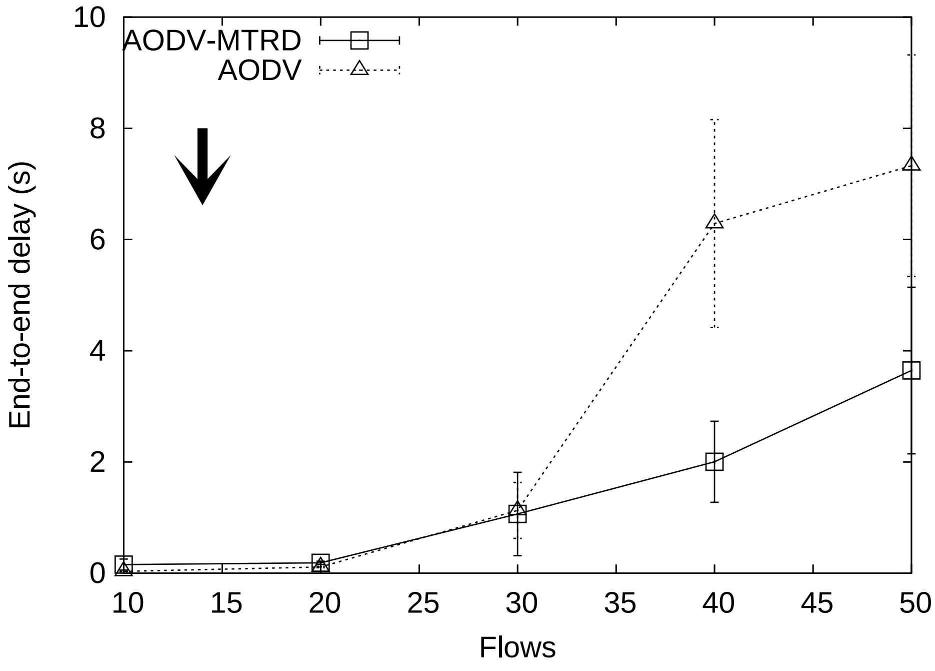


(a) Packet delivery ratio.

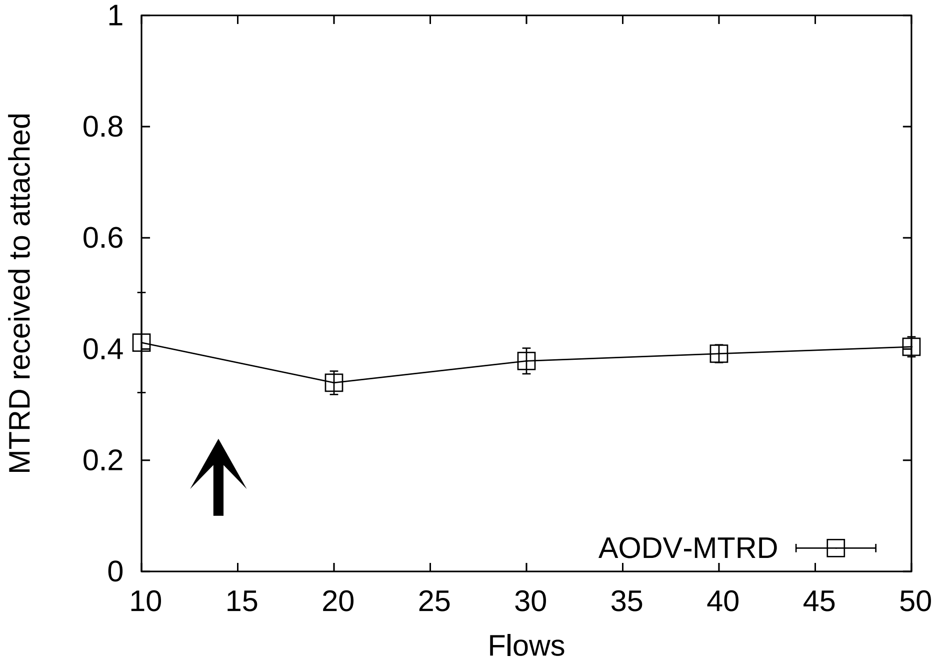


(b) Control overhead.

Figure 5.5: Performance when number of flows changes. 100 nodes, 1400×1400 m^2, node speed [0.1, 20]m/s, and pause time 30s.

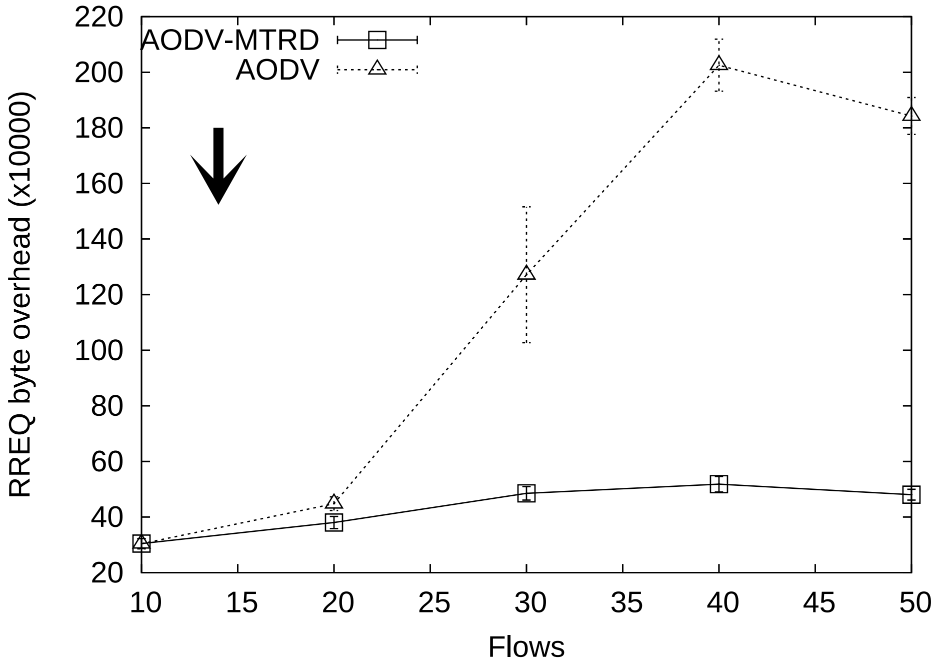


(c) End-to-end delay.

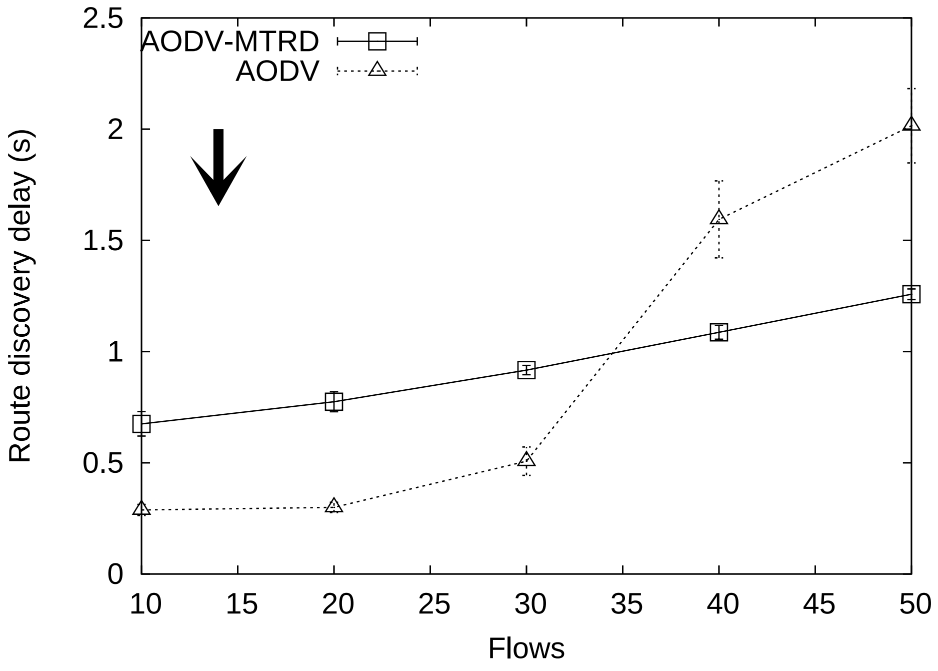


(d) MTRD received to attached.

Figure 5.5: Performance when number of flows changes. 100 nodes, $1400 \times 1400\ m^2$, node speed [0.1, 20]m/s, and pause time $30s$ (continued)



(e) RREQ byte overhead.



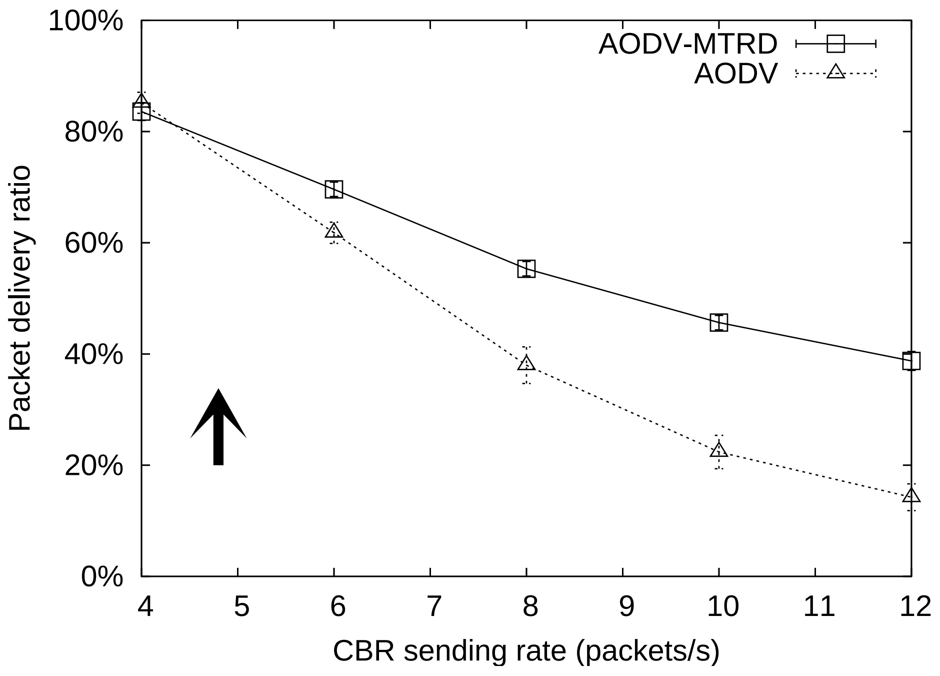
(f) Route discovery delay.

Figure 5.5: Performance when number of flows changes. 100 nodes, 1400×1400 m^2, node speed [0.1, 20]m/s, and pause time 30s (continued)

The use of MTRD may increase the route discovery delay, which is the interval between the time that a source originates a route request and the time that the source receives a valid route reply. If MTRD fails, the time spent on MTRD is wasted and contributes to the route discovery delay. Figure 5.5(f) shows the route discovery delay of AODV-MTRD and AODV. We observe that when the number of flows is low, AODV-MTRD has more route discovery delay than AODV, because MTRD finds less attaching opportunities. However, from Figure 5.5(c), AODV-MTRD still has about the same per-packet end-to-end delay as AODV. When the number of flows increases, AODV-MTRD has shorter route discovery delay.

Varied CBR Rate and Varied Maximum Node Speed

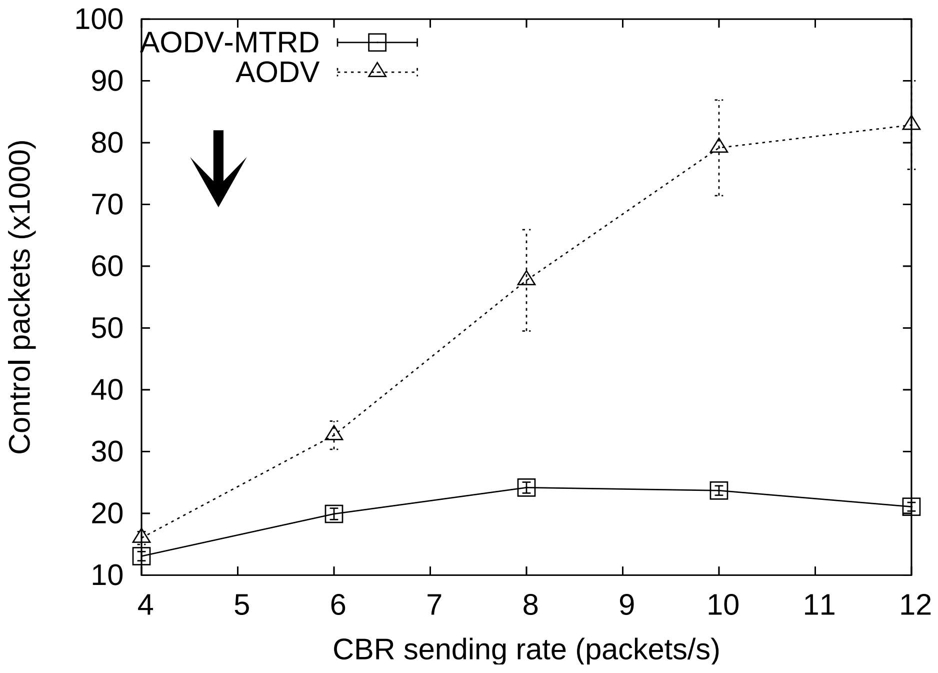
Figures 5.6(a), 5.6(b), and 5.6(c) show the results of PDR, control overhead per flow, and end-to-end delay, respectively, when the CBR sending rate varies from 4 to 12 packets per second. We observe that MTRD considerably improves the performance of AODV for all metrics, especially when sources send data packets at a higher rate.



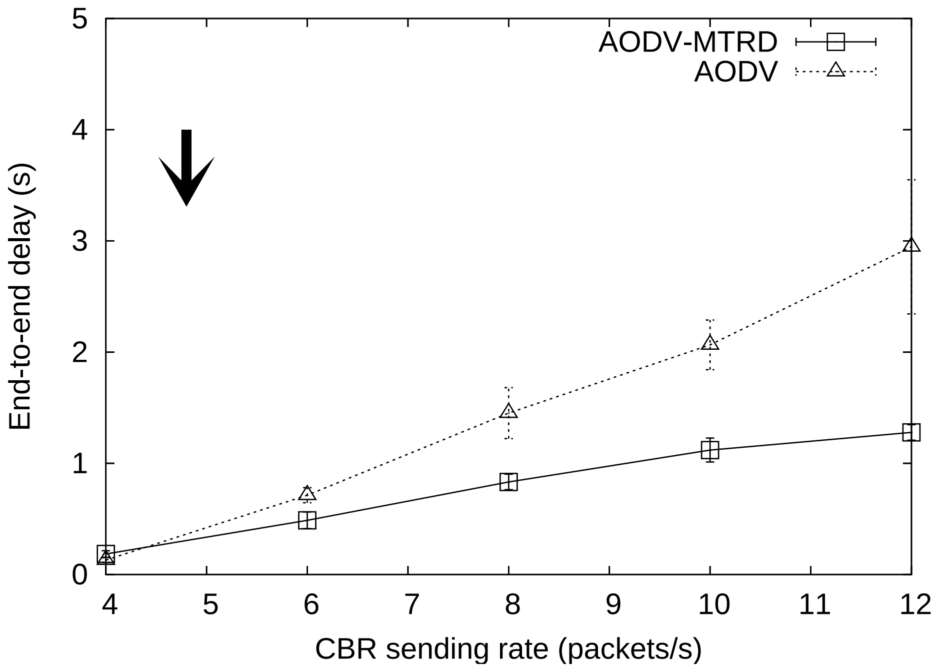
(a) Packet delivery ratio.

Figure 5.6: Performance when CBR rate changes. 100 nodes, 1400×1400 m^2, 20 flows, node speed [0.1, 20]m/s, and pause time 30s.

Figures 5.7(a), 5.7(b), and 5.7(c) show the results of PDR, control overhead per flow, and end-



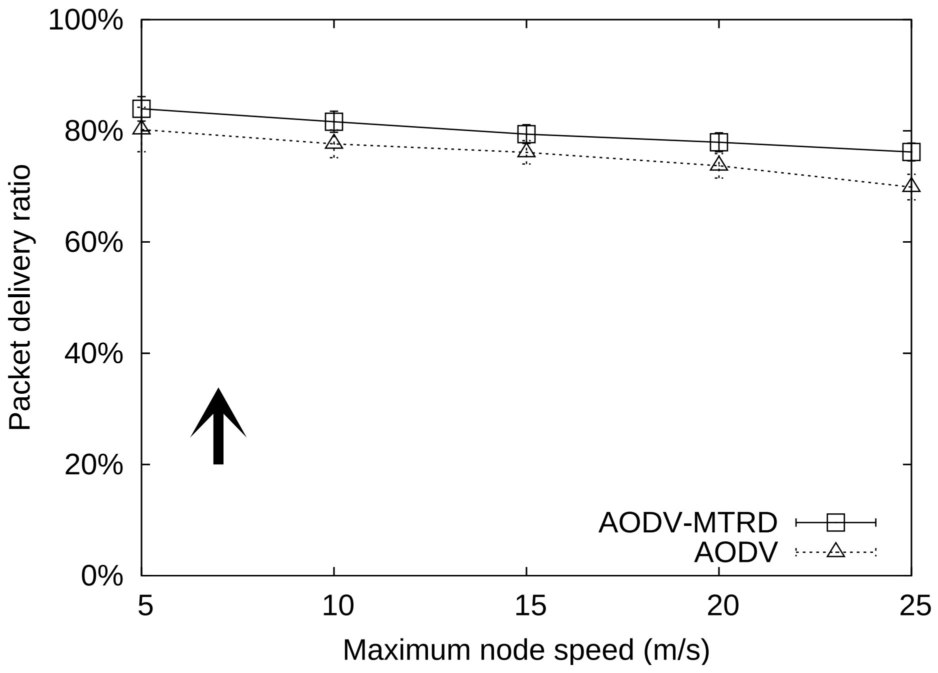
(b) Control overhead.



(c) End-to-end delay.

Figure 5.6: Performance when CBR rate changes. 100 nodes, $1400\times1400\ m^2$, 20 flows, node speed [0.1, 20]m/s, and pause time $30s$ (continued)

to-end delay, respectively, when the maximum node speed varies from 5 to 25 m/s. We observe that AODV-MTRD reduces the control overhead significantly, while the PDR and the end-to-end delay are comparable.



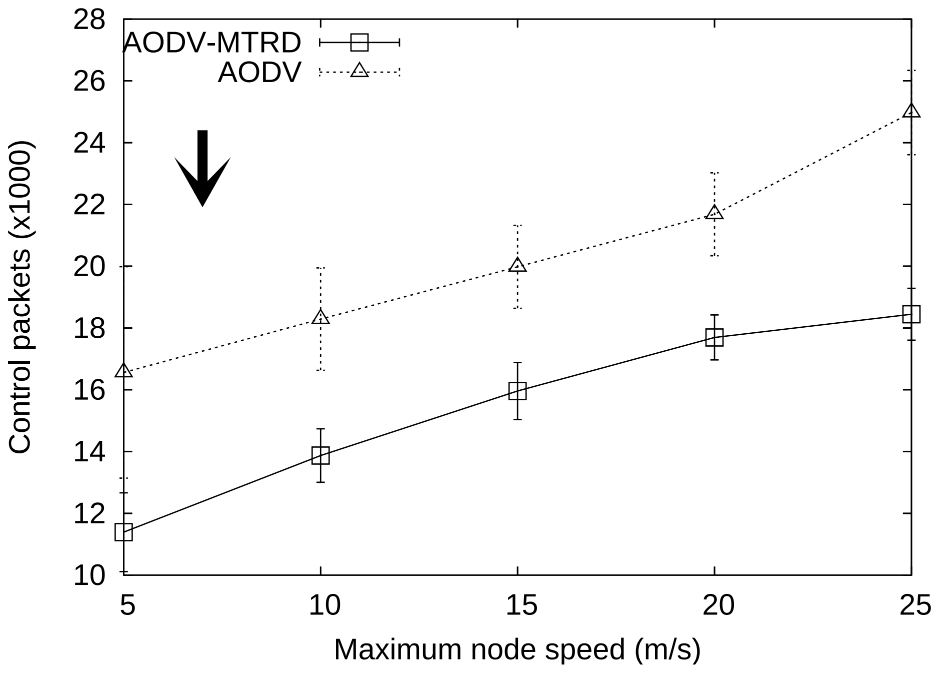
(a) Packet delivery ratio.

Figure 5.7: Performance when max node speed changes. 100 nodes, 1400×1400 m^2, 25 flows, min speed $0.1m/s$, and pause time $0s$.

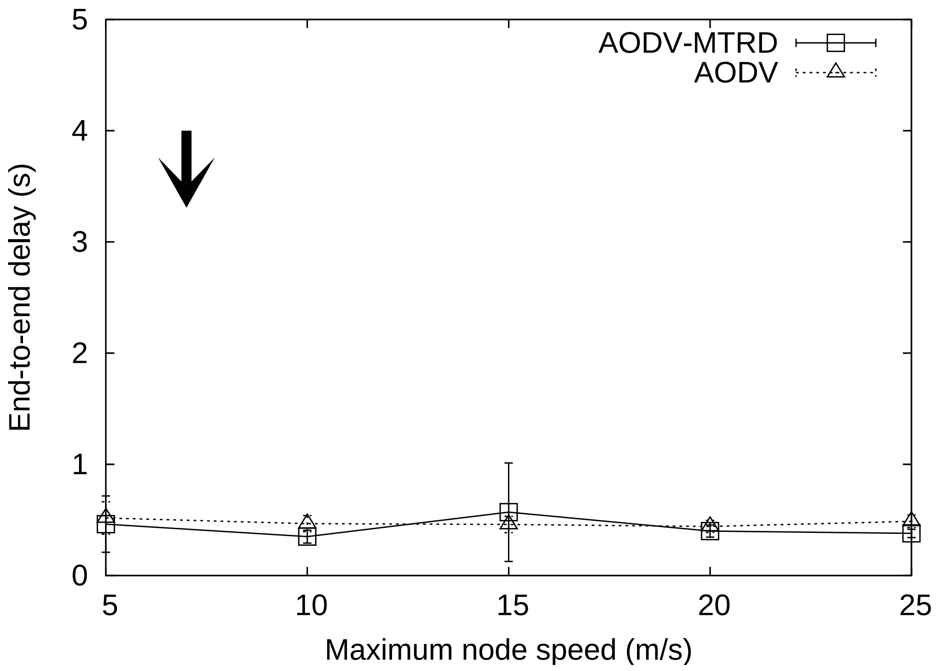
The reasons for the improvements are similar to the previous analysis in Section 5.4.2. When more route discoveries need to be conducted due to increasing data traffic or topology changes, MTRD finds more opportunities to attach route requests. Consequently, the control overhead is reduced and other metrics also improve.

Varied Network Size

Figures 5.8(a), 5.8(b), and 5.8(c) show the results of PDR, control overhead, and end-to-end delay, respectively, when the network size varies from 100 nodes to 300 nodes. We observe that the performance of AODV degrades rapidly when network size increases. Nevertheless, MTRD improves the performance of AODV significantly. For example, when the network size is 150, AODV-MTRD improves the PDR from 10% to 52% (Figure 5.8(a)), reduces the control packets by 72% (Figure 5.8(b)), and reduces the end-to-end delay by 69% (Figure 5.8(c)).

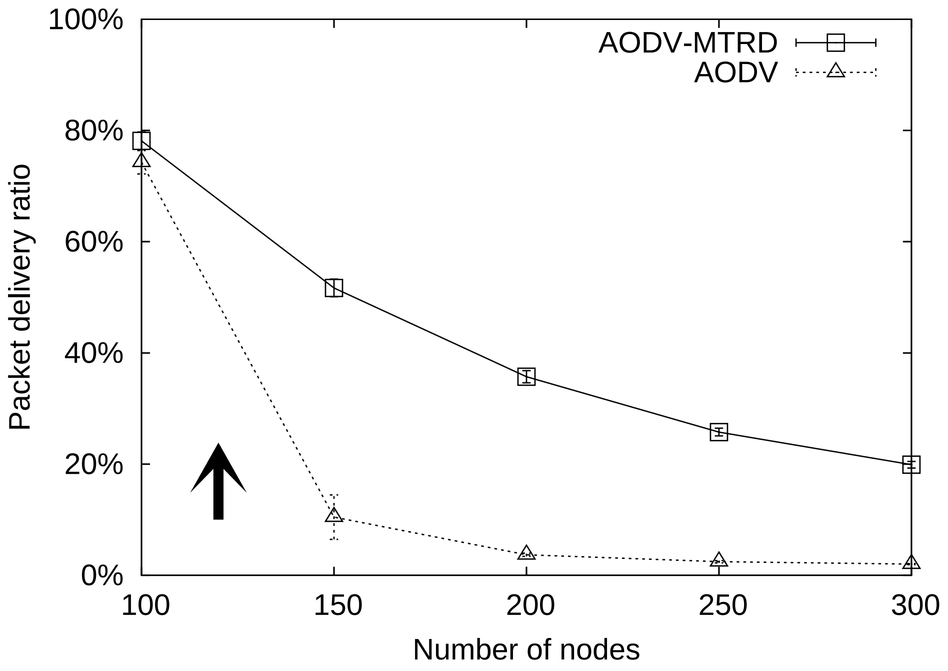


(b) Control overhead.

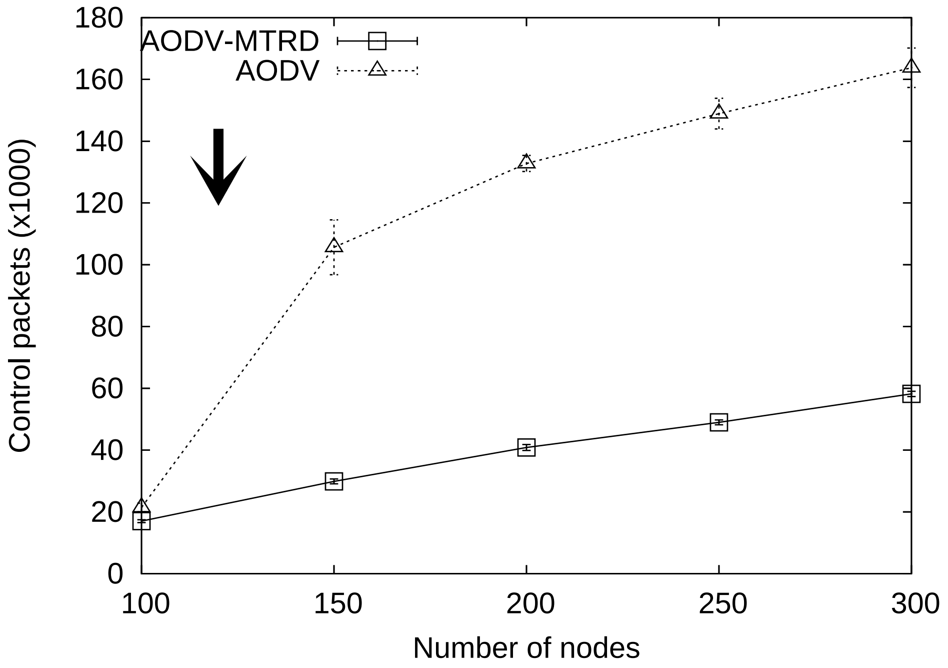


(c) End-to-end delay.

Figure 5.7: Performance when max node speed changes. 100 nodes, 1400×1400 m^2, 25 flows, min speed $0.1 m/s$, and pause time $0s$ (continued)

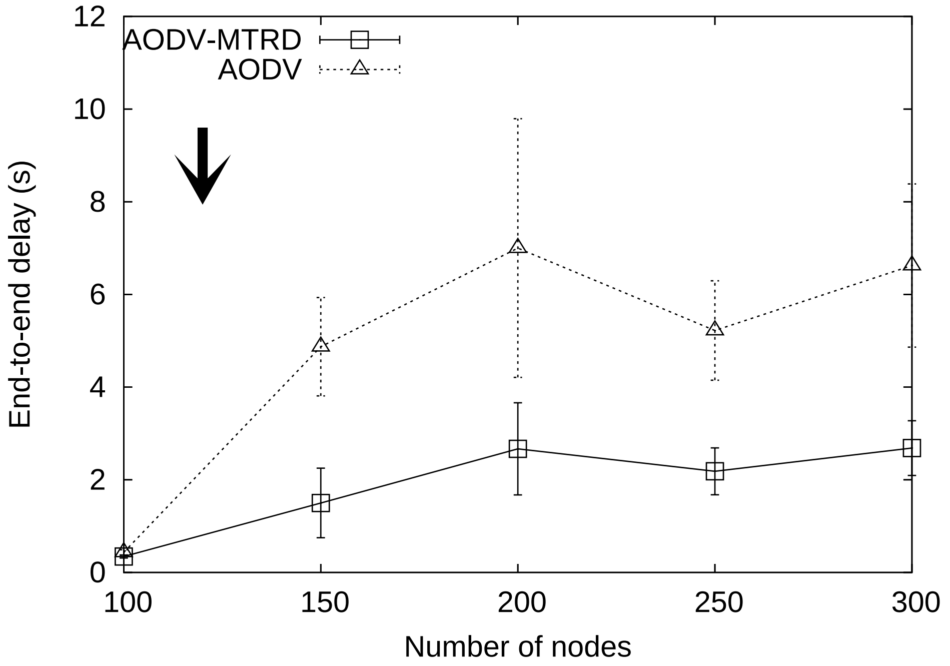


(a) Packet delivery ratio.



(b) Control overhead.

Figure 5.8: Performance when network size increases. Flows 25% of number of nodes. Node speed [0.1, 20]m/s, and pause time 30s.



(c) End-to-end delay.

Figure 5.8: Performance when network size increases. Flows 25% of number of nodes. Node speed [0.1, 20]m/s, and pause time 30s (continued)

When the network size increases, route discovery through flooding becomes more expensive, i.e., it requires more RREQ packets and longer delay. Moreover, routes in larger networks are longer and break more often. Thus, more route discoveries have to be conducted. MTRD finds more opportunities to attach route requests and improves routing performance significantly.

5.5 Summary

Route request (RREQ) packets dominate the control overhead in on-demand routing protocols for mobile ad hoc networks (MANETs). Usually, a considerable number of RREQ packets are in transit in the network. A node cooperatively relays RREQs for other nodes at a high rate. These RREQs provide opportunities for intermediate nodes to discover routes for their own sake.

In this chapter, we exploited the idea of carpooling in MANETs to optimize routing performance. We designed a new route discovery scheme called multiple-target route discovery (MTRD). In MTRD, when a source needs to discover a route, instead of immediately broadcasting a new RREQ message, the node first tries to attach its request to the existing RREQ packets that it relays.

This approach can improve routing performance by reducing the control overhead, congestion, and power consumption.

We conducted extensive simulations to evaluate the performance of MTRD. The results confirm the effectiveness of MTRD by showing that MTRD significantly reduces control overhead, improves the packet delivery ratio, and reduces the end-to-end delay when the network load is medium to heavy.

Chapter 6

Bilateral Route Discovery

6.1 Introduction

Traditionally, in on-demand routing protocols, the source node assumes most of the responsibility of discovering a route. The source determines parameters such as time-to-live (TTL) and waiting time, and broadcasts a RREQ (route request) message. The destination simply responds to a RREQ with a RREP (route reply) message. For this reason, on-demand routing protocols are also called source-initiated on-demand routing protocols [92]. We call the traditional manner of route discovery *unilateral route discovery (URD)*.

The unilateral operation is not balanced, because one party bears more burden than the other. It is not desirable and may have long delays. If route discoveries can be conducted in a more balanced way, the efficiency of on-demand routing is likely to be greatly improved. Thus, a natural question is: Can the destination participate in a route discovery more actively? This chapter answers this question and proposes a new scheme called *bilateral route discovery (BRD)* for on-demand routing protocols. BRD has potential to improve routing performance by reducing control overhead and route discovery latency. The main contributions of this chapter are:

(i) We address the disadvantage of traditional route discovery that operates in a unilateral manner, and propose BRD, in which both source and destination actively participate in route discovery process.

(ii) As an underlying protocol for BRD, we propose gratuitous route error reporting (GRER). GRER uses a relaying node to bypass the failed link and notifies the destination of a broken route. The destination can thus actively participate in the upcoming route re-discovery process.

(iii) We study the performance of BRD by simulating it in conjunction with AODV. The results show that BRD significantly improves routing performance.

The rest of the chapter is organized as follows. In the next section, we discuss the motivation and challenges for BRD. The GRER and the BRD protocols are presented in Section 6.3 and Section 6.4, respectively. Section 6.5 presents simulation setup and results. We summarize this chapter in Section 6.6.

6.2 Motivation and Challenges

A number of researchers [42, 50, 94] have studied the duration of links and paths in MANETs. In summary, path duration under mobility models such as *random waypoint* [23] and *reference point group mobility* [51] is exponentially distributed under a few reasonable conditions[1]. The probability density function can be written as $f(x) = \lambda e^{-\lambda x}$. The parameter λ depends on hop count h, node speed v, and transmission range R [94]. Simulation results [94] show that at the connectivity level (based on the network topology recorded in mobility trace files and a shortest path algorithm), path duration ranges from 5 to 12 seconds and the average is about 7 seconds. If measured at the routing level, i.e., the duration of paths discovered by a routing protocol, the result is lower because: first, paths discovered by the routing protocol may be suboptimal; second, congestion on communication channels may cause route breakages. Since routes in MANETs break and need to be re-discovered often, efficient route discovery schemes are desirable.

On-demand routing protocols discover a route by *flooding* the entire network or a part of it with a RREQ message. A TTL field in the RREQ message limits its propagation scope and thus avoids network-wide flooding. The advantage of flooding is its simplicity, whereas the disadvantage is that it is unbalanced and may incur considerable overhead, especially when the network load is heavy or mobility is high. We investigate discovering routes in a more balanced way, i.e., both the source and the destination actively participate in a route discovery.

6.2.1 Motivation

To illustrate, we introduce the notion of *request zone*. The request zone of a route discovery is a zone that is expected to cover the destination. Nodes within the request zone receive a RREQ message for this route discovery. This definition is different from the definition in LAR (location aided routing) [64], where nodes in a request zone not only receive but also forward the request. Defining a request zone does not necessarily indicate that BRD is a position-based approach.

The request zone of URD is a circle centered at the source, with the radius not less than the distance from the source to the destination (denoted as r). Figure 6.1 shows the request zone of URD (the dashed-line circle).

If the destination participates in discovering the route, the search space can be depicted by two

[1]E.g., maximum node speed $\geq 10m/s$, hop count ≥ 2 [94].

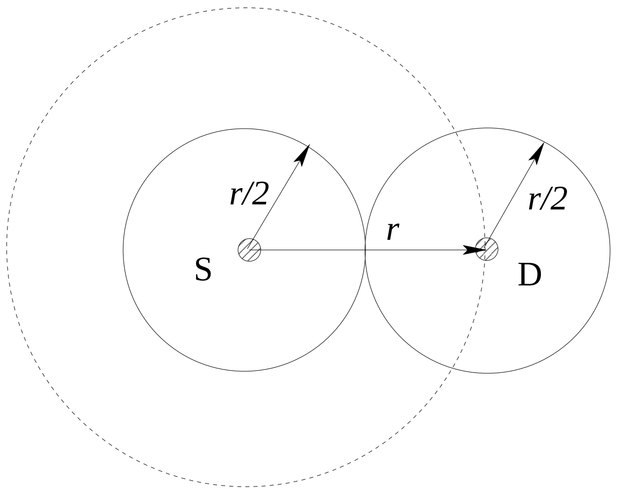


Figure 6.1: Request zones of URD and BRD.

smaller circles (solid line in Figure 6.1): one centered at the source (*source circle* or $\mathcal{C}_s$) and the other centered at the destination (*destination circle* or $\mathcal{C}_d$). When $\mathcal{C}_s$ and $\mathcal{C}_d$ intersect and some intermediate nodes are located in the intersection, a route is likely to be established between the source and the destination. We call these nodes *intersection nodes*.

The destination generates $\mathcal{C}_d$ by initiating its own route discovery, searching for a path from the destination to the source. Thus, BRD consists of two halves: a *source route discovery* or srd (initiated by the source and searching for the destination) and a *destination route discovery* or drd (initiated by the destination and searching for the source). We now analyze the optimal radii of $\mathcal{C}_s$ and $\mathcal{C}_d$, denoted as $\mathcal{R}_s$ and $\mathcal{R}_d$, respectively.

Let x be the variable for $\mathcal{R}_s$, then $\mathcal{R}_d = r - x$. The area ($\mathcal{A}_{brd}$) of the request zone is $\mathcal{A}_{brd} = \pi x^2 + \pi(r-x)^2$. $\mathcal{A}_{brd}$ reaches its minimum when $d\mathcal{A}_{brd}/dx = 0$, which occurs at $x = r/2$. Thus, the optimal values of $\mathcal{R}_s$ and $\mathcal{R}_d$ are one half of the distance between the source and the destination, and the minimum value of $\mathcal{A}_{brd}$ is $\pi(r/2)^2 * 2 = \pi r^2/2$. On the other hand, URD has a request zone of area $\mathcal{A}_{urd} = \pi r^2$. Therefore, BRD may incur as little as a half of the overhead of URD.

In real networks, the performance gain of BRD might be less for a number of reasons. First, we did not consider the network boundary in the analysis above. The overhead of URD is smaller if its request zone is cut by the network boundary. Second, the request zone is a geometric estimate, whereas routing protocols typically use hop count to control the propagation of route requests. The geometric estimate may not be very precise. Third, to increase the probability of a successful BRD, $\mathcal{C}_s$ and $\mathcal{C}_d$ should overlap somewhat. This overlap increases the overhead of BRD. Finally, BRD may not always succeed. For example, it is possible that no intersection nodes exist, or transmissions for srd and drd collide at intersection nodes, as a result of the hidden-terminal problem [98].

The BRD approach has two main advantages. First, BRD has the potential to reduce control overhead by about half compared to URD. Lower control overhead means lower congestion and lower power consumption. Second, BRD operates in a more balanced manner than URD. The overhead is evenly distributed around the source and the destination, which leads to a more uniform consumption of resources and mitigates congestion in the network.

6.2.2 Challenges

BRD poses a number of design challenges. We first discuss how a destination can be notified to initiate a drd, and then discuss challenges in conducting a BRD.

Symmetric Route Error Reporting

When a source discovers a route to a destination for the first time, the destination is not able to actively participate in the discovery, because it does not know about the upcoming communication, unless communication is scheduled in advance. However, after the first successful route discovery using URD, the destination is aware of the communication and is able to provide assistance in future route re-discoveries.

In existing on-demand routing protocols, when an active route breaks at a link, the upstream node U of the failed link learns about the failure when it transmits a data packet to the downstream node V of the failed link but receives no acknowledgment[2]. Then U sends a RERR (route error) message to the source S of the data packet.

Upon receiving the RERR message, S learns that the route is broken. However, the destination D is not aware of the break. We call this problem the *asymmetric route error reporting* problem; it exists in most on-demand route protocols that do not use periodic hello messages.

This problem is particularly important for BRD, because the destination has to be notified of a route breakage before it can actively participate in the upcoming route discovery. Designing such a route error reporting scheme is challenging, because when U detects a link failure, V (and therefore D) is already unreachable.

[2]The acknowledgment can be implemented in three ways: link-layer, passive (listening the forwarding by the next-hop node), and network-layer [59].

Conducting Bilateral Route Discovery

After both S and D are notified of the route error, they are ready to conduct a BRD. The following are details they must attend to:

(i) The TTL settings of srd and drd. The TTLs should be large enough to ensure the existence of intersection nodes, but not too large to compromise the efficiency of BRD.

(ii) Sending cached RREPs[3]. The number of intersection nodes may be large, so many intermediate nodes are able to send a cached RREP to the source. It is not desirable for each intersection node to send a cached RREP to the source, or, the channel in the intersection area may become congested. Unlike RREQ packets, RREP packets are transmitted using unicast. In a MAC layer protocol such as IEEE 802.11 [7], unicast transmission is expensive because it requires RTS/CTS, data, and ACK frames. Thus, the number of cached RREPs should be controlled.

(iii) The receiving order of RREQ messages. For an intersection node I, it is better to receive a RREQ message from drd first and learn about a route to the destination. Then, when I receives a RREQ message for srd, I can send a cached RREP to the source. However, intersection nodes may receive RREQ messages from srd first. A buffer mechanism is required to increase the probability of a successful BRD.

6.3 Gratuitous Route Error Reporting

BRD consists of two halves. The srd half works like the existing URD scheme, but the drd half is challenging because existing routing protocols lack a mechanism to notify the destination of a broken route. We could implement notification in several ways.

First, destination D may assume a route break when it receives no data packets for some time. This approach is simple, but selecting a proper timeout value is difficult. The topology of MANETs changes dynamically. A small timeout may cause many false positives (D assumes a route is broken, but in fact it is not); a large timeout compromises the efficiency of BRD.

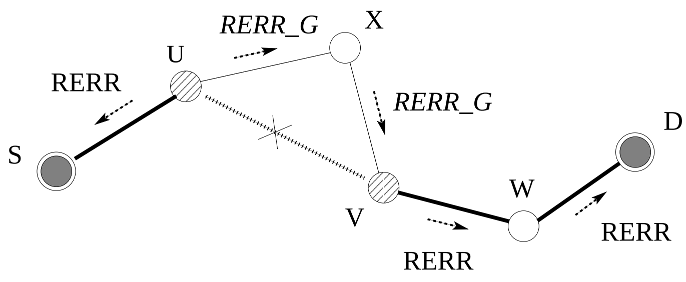
Second, when source S starts to discover a route to D, S may signal D through a separate control channel. S and D first contact each other through a low-bandwidth but long-range control channel, but then send data on multiple-hop channels with higher bandwidth. This approach makes a strong assumption about the radio channels, and the assumption may not always be reasonable.

[3]A *cached RREP* is generated by an intermediate node that knows a route to the destination.

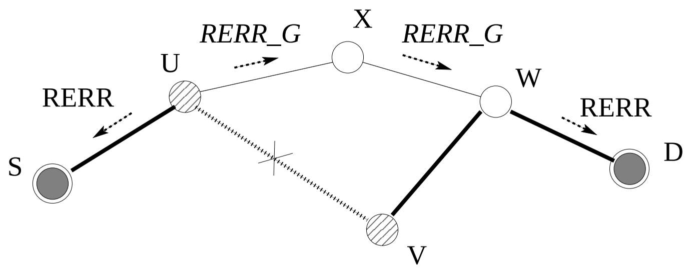
Third, U (the forwarding node upstream of the failed link) can send a RERR message to notify D of the route break. However, since V (the forwarding node downstream of the failed link) typically is not aware of the link break, we need a scheme for U to signal V or its successors on the broken route.

In this work, we use the third approach and propose *gratuitous route error reporting (GRER)*. The basic idea of GRER is that U broadcasts a gratuitous route error message (RERR_G) with a TTL of 2 to bypass the failed link and reach V or one of its successors on the route.

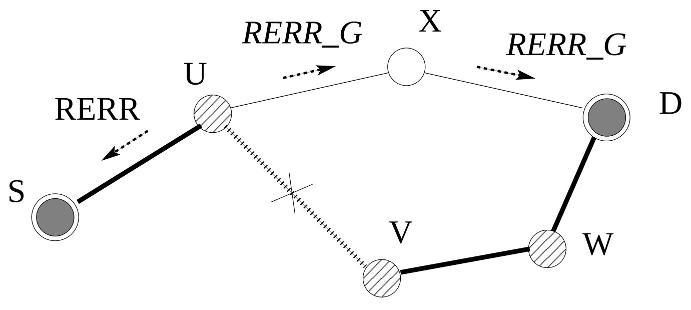
Figure 6.2(a) shows an example of how GRER works. When link UV fails, U broadcasts a RERR_G message. Node X relays (broadcasts) the message, and V receives the message. When V or other downstream nodes receive the message, they send a RERR message to the destination informing it of the route error.



(a) General case.



(b) Next hop on the route receives RERR_G.



(c) Destination receives RERR_G.

Figure 6.2: The handling of RERR_G message (three cases).

In the following, we discuss sending and handling a RERR_G message and discuss several design issues of GRER.

6.3.1 Sending Gratuitous Route Error

When link UV fails, U sends a RERR message to S. After that, U broadcasts a gratuitous route error message (RERR_G). A RERR_G message consists of fields indicating U, V, S, and D. It may also contain information as required by a particular routing protocol. For example, when used in AODV, a RERR_G message also needs to carry the sequence number of S, because AODV uses the sequence number to determine the age of a route.

Transmitting RERR_G messages may pose considerable extra overhead, because routes in MANETs may break often. Therefore, we use two approaches to reduce the overhead of RERR_G messages. First, we set the TTL of a RERR_G message to 2. It is very likely that the RERR_G reaches a downstream node after being relayed once (e.g., Figure 6.2(a)). A larger TTL would increase overhead with only small additional payoff. Second, we control the number of neighbors that relay the message, as discussed next.

6.3.2 Relaying a RERR_G Message

Typically, a RERR_G message is relayed one more hop after U originates it. On its second hop, the RERR_G reaches V with high probability, and V subsequently sends a RERR message to the destination. Figure 6.3 shows the flowchart of handling a RERR_G message[4].

A node relays the message if two conditions hold. First, V is in its neighbor table. Second, no other neighbors have already relayed the message. Both conditions aim to reduce the overhead of RERR_G messages. The first condition avoids unnecessary relays. A RERR_G message would be more likely to reach the end node when the end node is in the neighbor table[5] of the relaying node. The second condition suppresses duplicate relays.

To relay a RERR_G message, a node sets a jitter timer that is uniformly and randomly chosen from a range. This jitter prevents neighboring nodes from relaying the message simultaneously. When the timer expires, if the node has not heard any relay by its neighbors, the node relays the

[4]The flowchart combines the handling of RERR_G message in both Section 6.3.2 and Section 6.3.3.
[5]Maintaining neighbor table is discussed in Section 6.3.4.

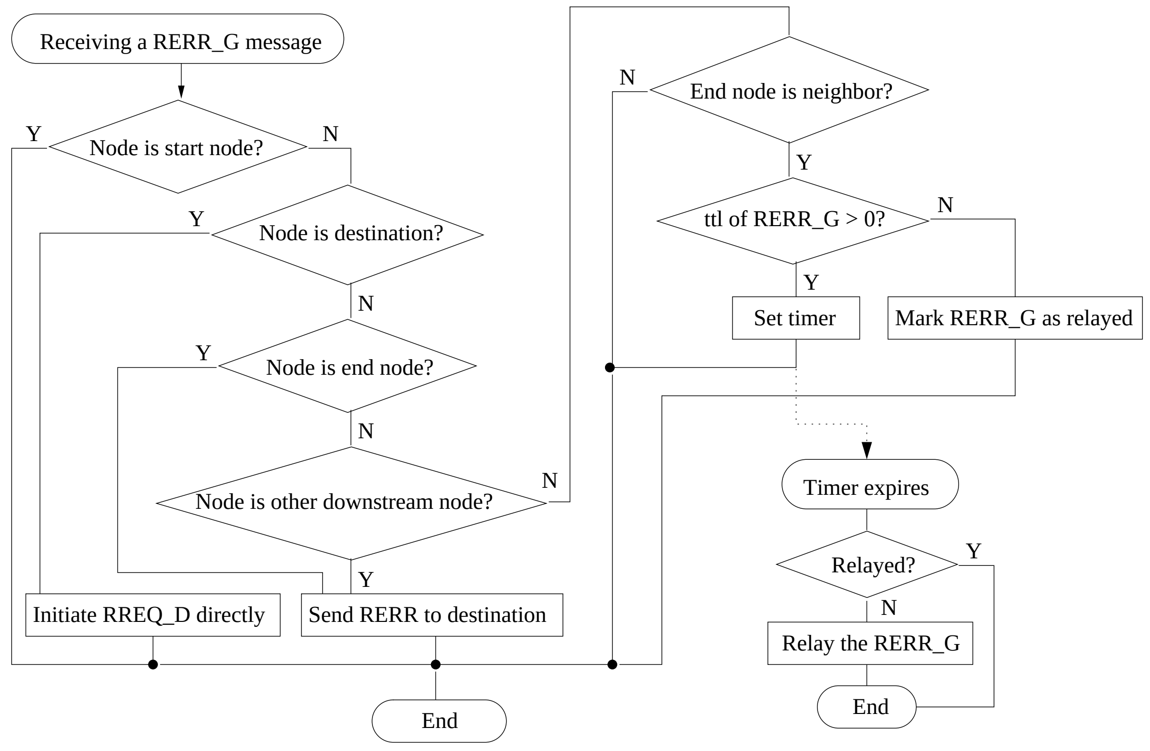


Figure 6.3: RERR_G message handling flowchart.

message. Thus, a RERR_G scheduled to be relayed earlier may suppress the RERR_Gs scheduled to be relayed later.

6.3.3 Notifying the Destination

When V or other downstream nodes on the broken route receive the RERR_G message, they send a RERR message to the destination informing it of the route error. The destination subsequently participates in a route discovery. The handling of the RERR_G message is classified into several cases based on the receiving node (e.g., end node or destination).

If the node is V, or one of its successors, such as W, the RERR_G message has successfully bypassed the failed link and reached a downstream node. In this case, the node sends a RERR message to the destination. In Figure 6.2(a), end node V sends a RERR after receiving a RERR_G. In Figure 6.2(b), node W sends a RERR to destination D because it is a downstream node on the broken route.

If the node is the destination, the node initiates a drd directly, as discussed in Section 6.4.

Figure 6.2(c) shows an example where a RERR_G message relayed by node X directly reaches destination D.

6.3.4 Discussions of GRER

We discuss two issues: maintaining a neighbor table utilizing RREQ messages, and a comparison between GRER and the gratuitous route error message in DSR.

Maintaining Neighbor Table

We do not use a hello protocol to maintain a neighbor table at nodes. A hello protocol may introduce significant overhead and may give an unwanted proactive flavor to the routing protocol. Instead, we utilize the RREQ messages received by nodes to build neighbor tables. This approach works well, because a topology change that breaks a route typically triggers a route discovery process. Nodes discover neighbors simply by checking the sender of a received RREQ packet.

A node deletes an entry in its neighbor table if it receives no RREQ message from that node for a certain amount of time. The node may also delete an entry based on a link layer notification.

Bai and Singhal show in their simulations that typically there is a large number of ongoing RREQ messages in a MANET [17]. For example, in a network with 100 nodes and 20 CBR (constant bit rate) flows[6], each node in the network transmits about 2 to 3 RREQ packets per second on average. This number of RREQ packets should be enough to maintain a neighbor table.

A Comparison to Piggybacking RERR Messages in DSR

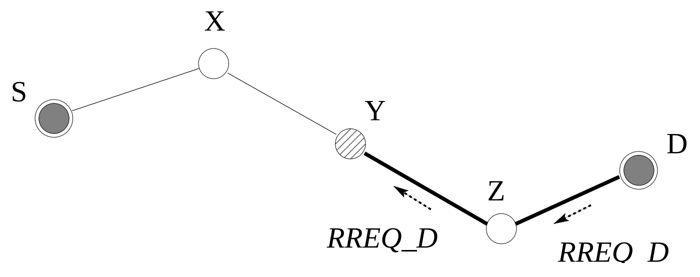
In DSR [59], when a source node receives a RERR message, it may piggyback the message on the RREQ message that it sends in response to the route error. Yu calls a RREQ with a RERR piggybacked a gratuitous route error message [107]. Thus, it may appear that both GRER and DSR use a "gratuitous route error". We use RERR_G for a fundamentally different purpose. It is used to fix a limitation in existing on-demand routing protocols, that a route error is only reported backward to the source, not to the destination. Thus, on average, about a half of a broken route is left intact, leaving a considerable amount of stale route information at nodes. With GRER, not only is stale route information purged, but the destination also learns about the route error and can actively participate in the ensuing route discovery.

[6]Other parameters: $1400\times1400m^2$, four 512-byte packets per second CBR traffic, min node speed $0.1m/s$, max node speed 5-20m/s, and pause time $30s$.

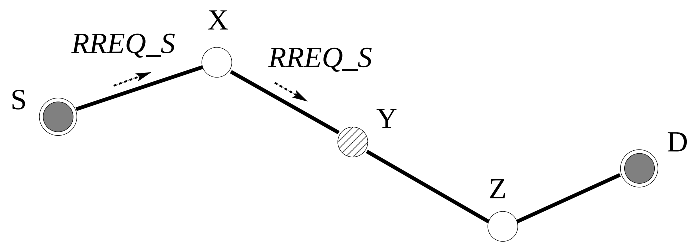
6.4 Bilateral Route Discovery (BRD)

After S and D are notified of a route breakage, they conduct a BRD, which consists of srd and drd components. Nodes covered by the drd learn a route to D, and nodes covered by the srd learn a route to S. Intersection nodes learn routes to both S and D, so they can send cached route replies to S. When S receives such a reply, BRD successfully discovers a route.

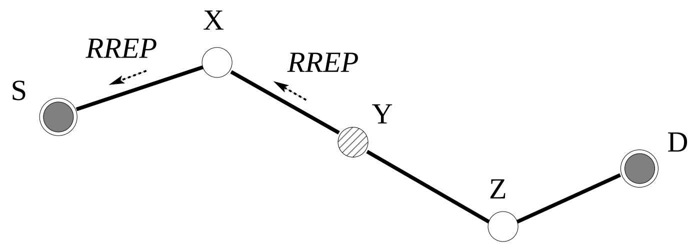
Figure 6.4 shows an example of BRD, where S is discovering a route to D. D initiates a drd and node Y learns a route to D. Similarly, S initiates a srd and Y learns a route to S. Y sends a cached route reply to S. When S receives the reply, BRD establishes a route from S to D.



(a) Destination route discovery.



(b) Source route discovery.



(c) Cached route reply

Figure 6.4: An example of BRD: (a) D initiates a drd, and Y learns a route to D. (b) S initiates a srd. Y is an intersection node and learns a route to S. (c) Y sends a cached RREP to S.

The operation of BRD involves initiating the route request and handling the route request and route reply. Next, we discuss these issues in detail.

6.4.1 Initiating *srd* and *drd*

When a route breaks, if the communication session between S and D is still active, both S and D initiate their route discoveries. D initiates a drd right after receiving a RERR or a RERR_G message reporting the route error. S initiates a srd typically after receiving a RERR message and when it has a data packet for the same destination and needs a route to send the packet.

Generally, a drd is conducted a bit earlier than its counterpart (srd). This timing allows intersection nodes to first learn a route to the destination and then to answer the srd with a cached route reply. Nevertheless, we must handle the situation in which the srd arrives first.

To differentiate RREQ messages for a srd and for a drd, we denote RREQ messages originated from S and from D as RREQ_S and RREQ_D, respectively. The major difference between a RREQ_S and a RREQ_D is that a RREQ_S needs to be answered whereas a RREQ_D does not.

BRD sets the TTL of a RREQ_S message (ttl_s) and the TTL of a RREQ_D message (ttl_d) according to the following equations:

$$\begin{aligned} ttl_s &= \lceil HC_{known}/2 \rceil \\ ttl_d &= \lfloor HC_{known}/2 \rfloor \end{aligned} \tag{6.1}$$

where HC_{known} is the hop count of a previously known route. Thus, ttl_s and ttl_d are determined based on two considerations: First, their sum is equal to HC_{known}, because the TTL of a route re-discovery in URD is typically set to the hop count of a previously known route. Second, they are roughly equal, to minimize the request zone, as discussed in Section 6.2.1.

6.4.2 Handling Requests and Reply

As in existing on-demand routing protocols, each node maintains a request-seen table to determine if a route request is duplicate. BRD utilizes both new and duplicate requests to maintain neighbor tables. However, BRD only uses new requests for route discovery[7].

Handling a RREQ_D Message

Upon receiving a new RREQ_D message, a node learns or updates a route to the destination. Typically, route update is based on two criteria: route age (use a new or newer route) and route quality (use a route with a better metric, e.g., lower hop count).

[7]One exception is DSR: A destination replies to each received RREQ. Thus, DSR can discover multiple paths.

If the source receives the message, it immediately learns a route to the destination. When an intermediate node receives the message, it checks if a new RREQ_S has been received, meaning the srd arrived first. If so, the node sends a reply to the source. We denote the message as RREP_I, where I represents for intermediate node.

If no new RREQ_S has been received, the node relays the RREQ_D message if the ttl_d is greater than zero. We call the nodes that receive a RREQ_D message with $ttl_d = 0$ the *d-perimeter nodes* of a drd. When a d-perimeter node receives a RREQ_S message later or has a saved RREQ_S message, it sends a RREP_I message. Other intersection nodes need not reply, to limit control overhead.

Figure 6.5 illustrates how d-perimeter nodes work. Source S and destination D are conducting a BRD, with both ttl_s and ttl_d set to 2. Nodes L, M, N, and O are d-perimeter nodes, and nodes M, N, P, and Q are intersection nodes. Only d-perimeter nodes M and N send RREP_I to S.

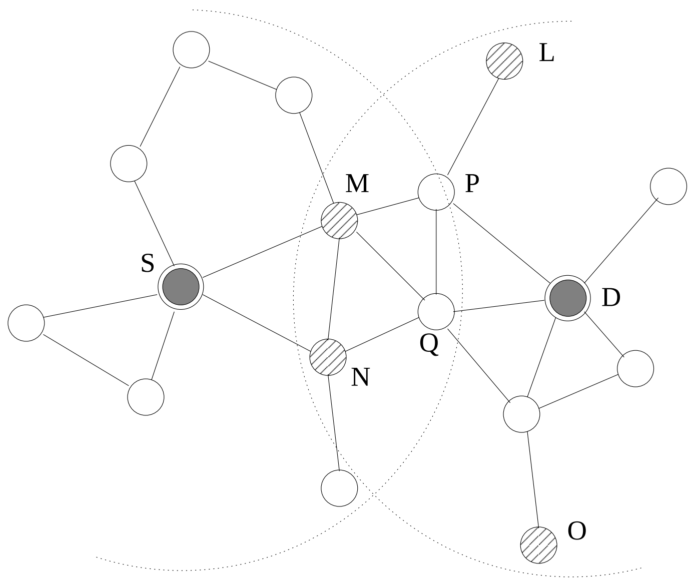


Figure 6.5: D-Perimeter nodes

Handling a RREQ_S Message

Upon receiving a new RREQ_S message, a node learns or updates a route to the source. If the destination receives the RREQ_S, meaning the request reaches the destination directly, it sends a RREP message to the source. When a d-perimeter node receives the RREQ_S, it sends a RREP_I message to the source. If the node cannot reply and the TTL of the RREQ_S is greater than zero, the node saves the request and relays the RREQ_S. Because the node saves the request (for some time), when the node receives a RREQ_D message and learns a route to the destination later, the node answers this request if it is a d-perimeter node.

Handling a RREP_I Message

Handling a RREP_I message is similar to handling RREP in existing on-demand routing protocols. Intermediate nodes on the route update their routing tables and relay the message to the source. The source transmits buffered data packets using the new route.

If the source does not receive any RREP or RREP_I message when the route discovery timer expires, it conducts a route discovery using traditional URD.

6.4.3 Unidirectional Links

The design of BRD discussed above implicitly assumes bidirectional links. This assumption is widely used in the majority of the routing protocols for MANETs, e.g., AODV, TORA [77], and proactive protocols based on distance vector routing. In MANETs, some links may be unidirectional because nodes may be heterogeneous and have different transmission ranges. Prakash [89] shows that using unidirectional links is costly in wireless ad-hoc networks. Marina and Das [70] show that there is almost no advantage in using unidirectional links. Thus, solutions to unidirectional links mainly focus on identifying and avoiding them. The proposed techniques include BlackListing [82], Hello [56], EUDA (early unidirectionality detection and avoidance) [65], and so on. If such a technique is used, a path is still bidirectional because only bidirectional links are used.

6.5 Performance Evaluation

We implemented BRD in AODV, calling the combination AODV-BRD, and conducted extensive simulations to evaluate its performance in comparison to AODV. Next, we discuss the simulation setup and then present the results and analysis.

6.5.1 Simulation Setup

We conducted simulations using GloMoSim 2.03 [21]. The settings for radio model and MAC layer are the same as described in Chapter 3.4. The traffic was Constant Bit Rate (CBR) and the mobility model was random waypoint [23]. Each simulation run lasted for 1200 seconds. The results in the graphs were averaged over 20 simulation runs, each with a different seed. Error bars represent the

95% confidence interval for each data point. We add a "goodness arrow" to each graph showing which direction is better. Unless specified otherwise, the following configurations were common: CBR sending rate four 512-byte packets per second, minimum node speed $0.1m/s$, maximum node speed $20m/s$, and pause time $30s$.

We first studied AODV-BRD and AODV in networks containing 100 nodes in an area of $1250 \times 1250m^2$ (Section 6.5.2). The number of flows varied from 10 to 30. We then studied AODV-BRD and AODV in networks of different sizes (Section 6.6). The number of nodes varied from 50 to 250. The number of flows was 20% of the number of nodes, and the node density was kept constant. Table 6.1 gives the details for this set of simulations.

Table 6.1: Network size, number of flows, and area.

Size	*Flows*	*Area (m^2)*
50	10	880^2
100	20	1250^2
150	30	1530^2
200	40	1770^2
250	50	1980^2

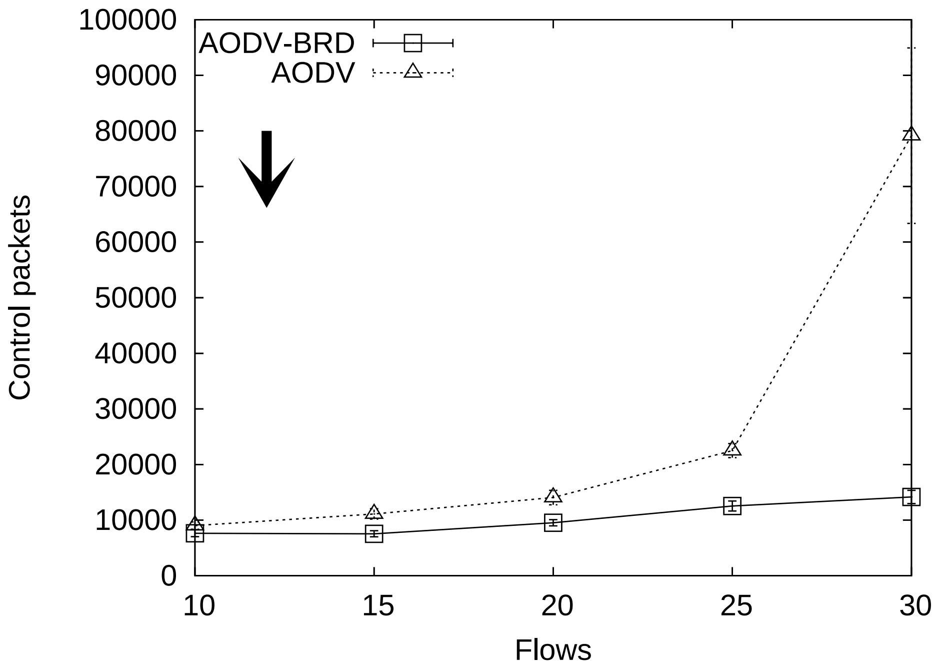
We measured three widely used metrics: *control overhead*, *packet delivery ratio (PDR)*, and *end-to-end delay*. Moreover, we measured three additional metrics for the 100-node simulations: BRD success ratio, GRER success ratio, and route discovery delay.

6.5.2 Results and Analysis

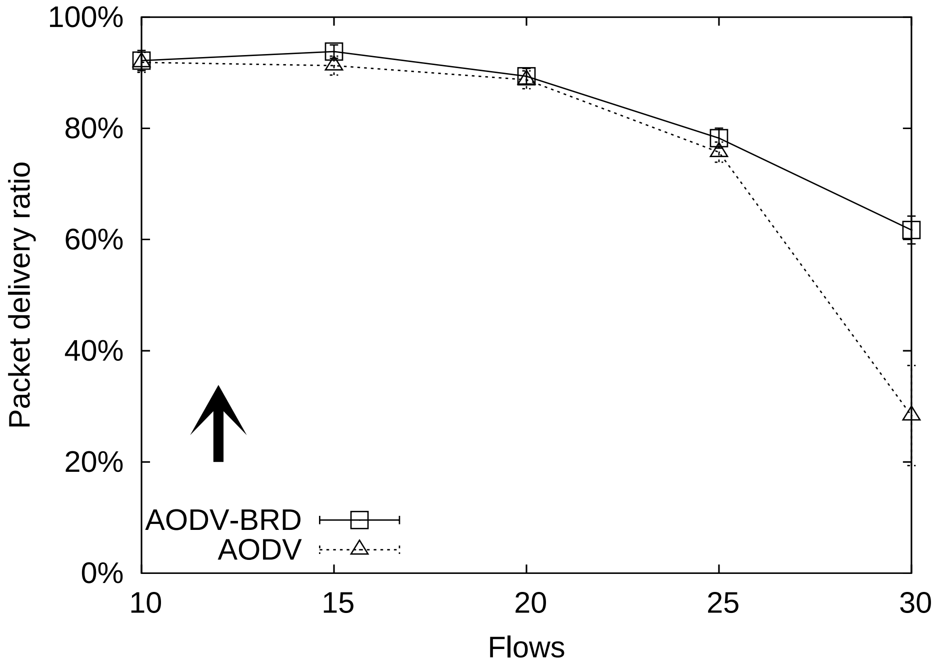
Varied CBR Flows

Figures 6.6(a), 6.6(b), and 6.6(c) show measured control overhead per flow, PDR, and end-to-end delay, respectively, when the number of flows varies from 10 to 30. We observe that with increasing number of flows, BRD improves the performance over AODV significantly. For example, when there are 30 flows, the control overhead is reduced by 80%, the PDR is improved by 118%, and the end-to-end delay is reduced by 65%.

These improvements justify our motivation. Compared to URD, BRD not only incurs much smaller overhead, but also works in a more balanced way. Therefore, control overhead is reduced, and congestion is mitigated. As a result, fewer packets are dropped due to collisions or queuing timeout, and data packets experience less queuing delay.

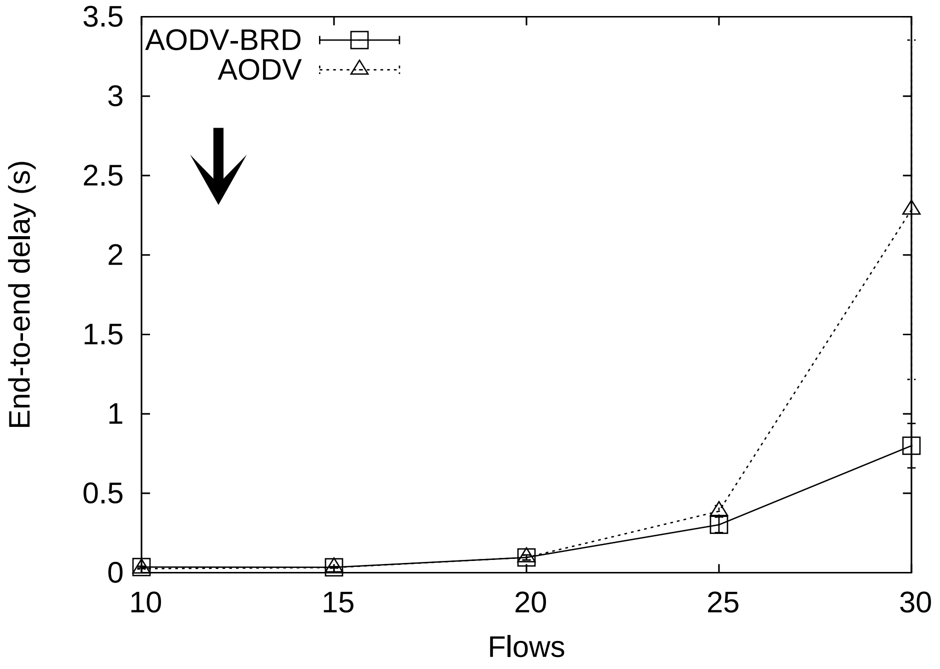


(a) Control overhead.

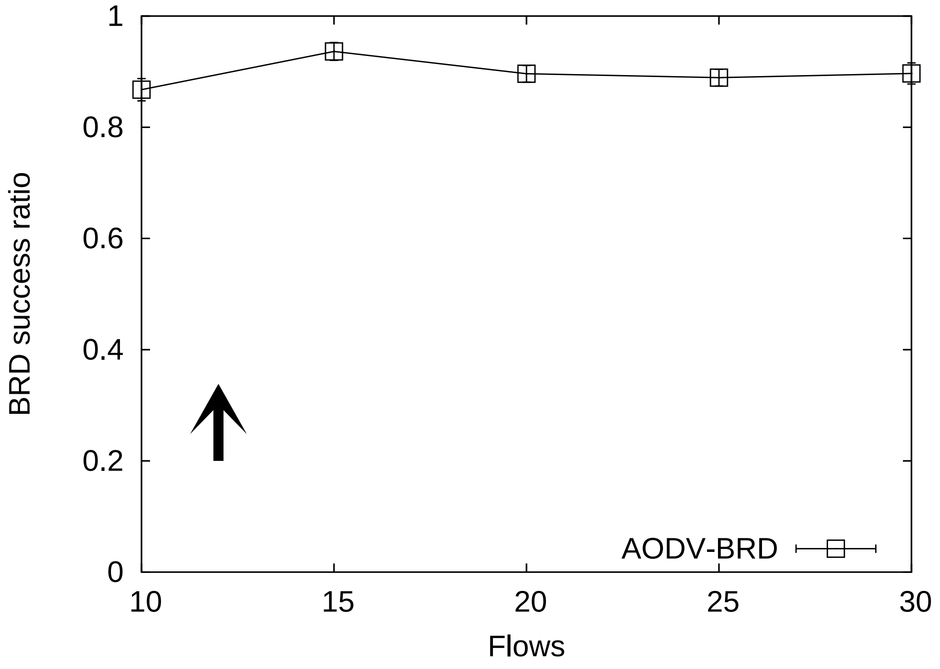


(b) Packet delivery ratio.

Figure 6.6: Performance when number of flows changes. 100 nodes in an area of 1250×1250 m^2.

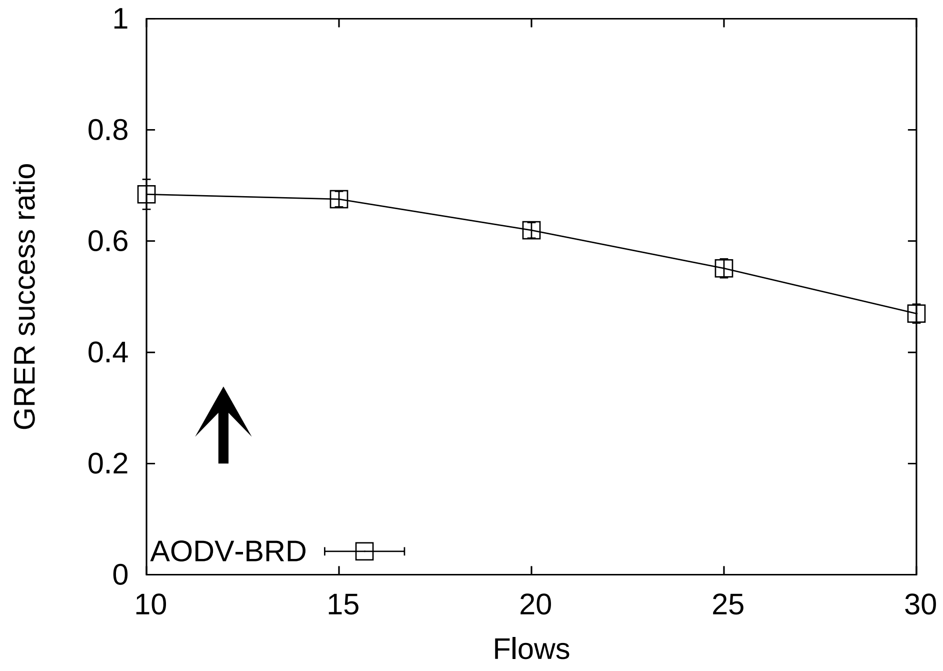


(c) End-to-end delay.

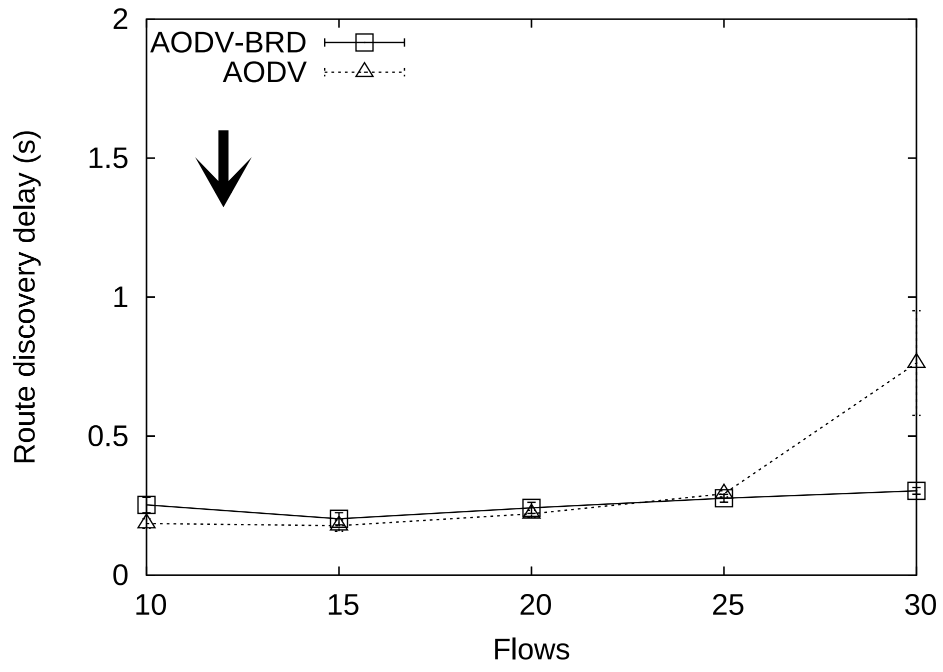


(d) BRD success ratio.

Figure 6.6: Performance when number of flows changes. 100 nodes in an area of $1250\times1250\ m^2$ (continued)



(e) GRER success ratio.



(f) Route discovery delay.

Figure 6.6: Performance when number of flows changes. 100 nodes in an area of 1250×1250 m^2 (continued)

Figure 6.6(d) shows the success ratio of BRD, i.e., the number of routes discovered by BRD divided by the number of times sources and destinations conduct srd and drd. We observe that the ratio is very high (about 90%), which indicates a BRD succeeds most of the time. Figure 6.6(e) shows the success ratio of GRER, i.e., the number of times the destination is notified of a route breakage divided by the number of times the start node initiates a GRER. Generally, about 50-70% GRER attempts succeed. The GRER success ratio decreases with the number of flows because more RERR_G messages are lost when more traffic is present in the network.

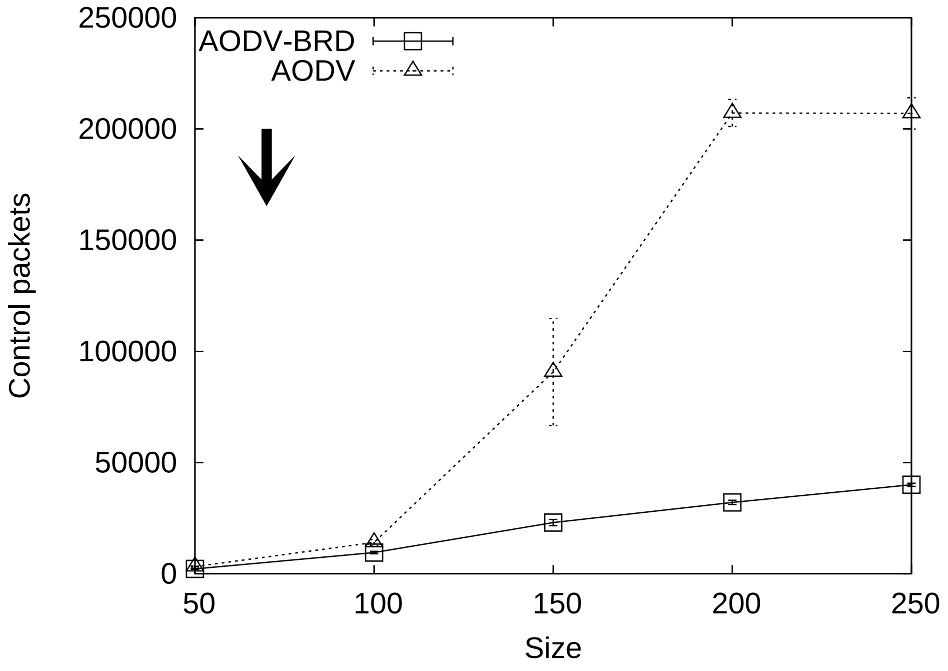
Figure 6.6(f) shows the route discovery delays of AODV-BRD and AODV. We observe that in general AODV-BRD and AODV have very comparable route discovery delays. When the number of flows is 30, the route discovery delay of AODV-BRD is about half that of AODV. BRD reduces the route discovery delay, but unsuccessful GRERs increase the delay.

Varied Network Size

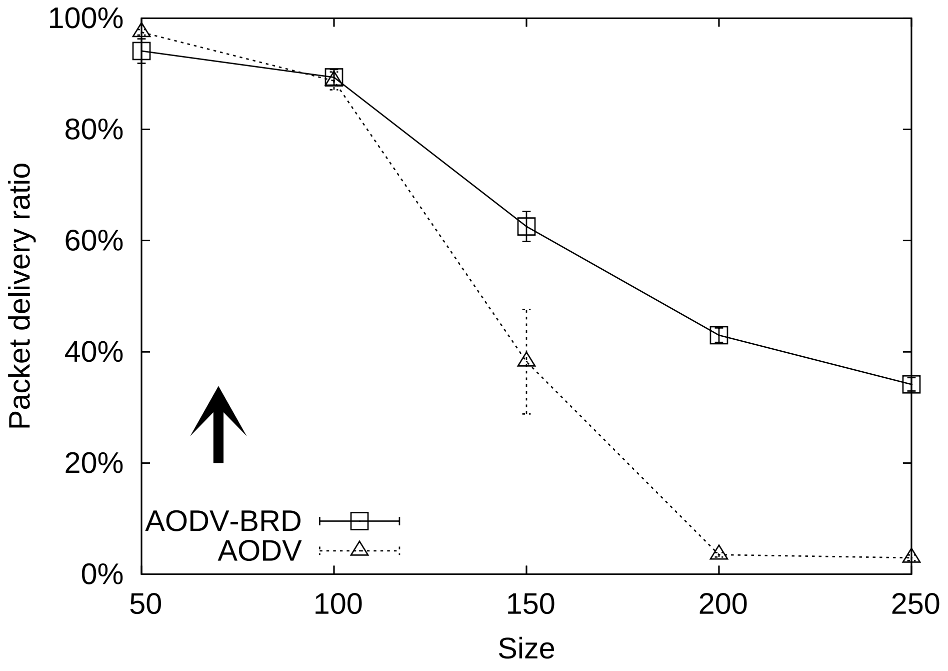
Figures 6.7(a), 6.7(b), and 6.7(c) show the results of control overhead per flow, PDR, and end-to-end delay, respectively, when the network size varies from 50 nodes to 250 nodes. We observe that the performance of AODV degrades rapidly when the network size increases. However, BRD improves the performance over AODV significantly. For example, when the network size is 150, AODV-BRD saves 75% of the control packets (Figure 6.7(a)), improves the PDR from 38% to 63% (Figure 6.7(b)), and reduces the end-to-end delay by 43% (Figure 6.7(c)).

6.6 Summary

To address the shortcomings of traditional unilateral route discovery (URD) for MANETs, we have proposed the idea of bilateral route discovery (BRD) where both source and destination actively participate in a route discovery. BRD might incur as little as a half of the overhead of URD. Moreover, BRD operates in a more balanced way than URD because it distributes overhead evenly around the source and the destination. As an underlying protocol for BRD, we have proposed the gratuitous route error reporting (GRER) to bypass the failed link and notify the destination of a broken route. The destination could thus participate in the BRD. We have conducted extensive simulations to evaluate the performance of BRD. The results show that BRD improves routing performance significantly.

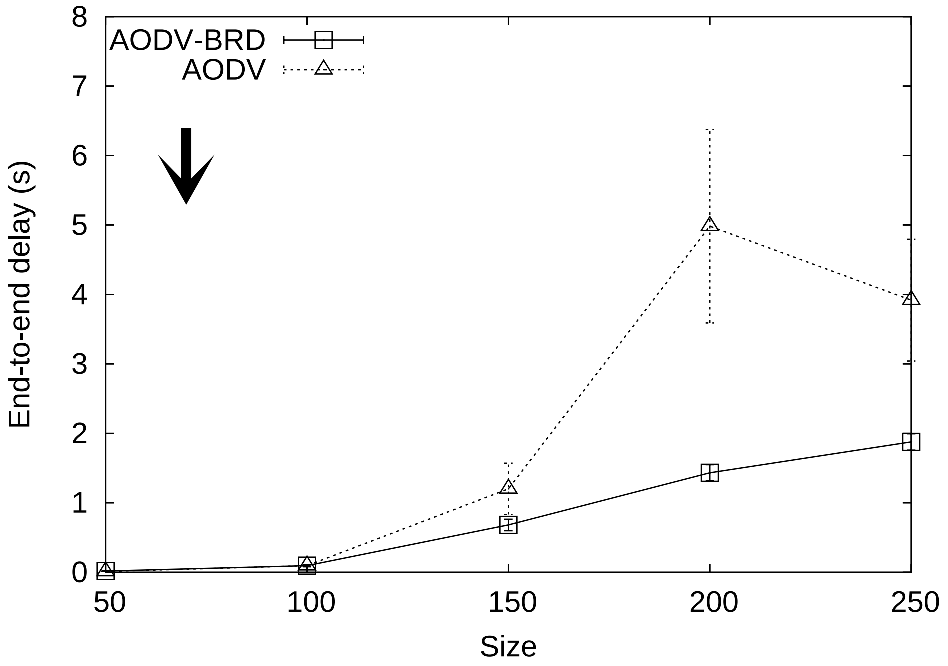


(a) Control overhead.



(b) Packet delivery ratio.

Figure 6.7: Performance when network size increases. The number of flows is 20% of number of nodes.



(c) End-to-end delay.

Figure 6.7: Performance when network size increases. The number of flows is 20% of number of nodes (continued)

Chapter 7

Supporting High-Throughput Routing Metrics

7.1 Introduction

Researchers have proposed a number of metrics to select routes with high throughput in multi-hop wireless networks. De Couto et al. [31] proposed the ETX (expected transmission count) metric. The ETX of a link is the expected number of transmissions to send a packet over that link, and the ETX of a path is the sum of the ETX of each link on the path. Draves et al. [34] proposed the ETT (expected transmission time) metric to find high throughput paths in heterogeneous, multi-radio WMNs. In this chapter, we collectively call metrics that select routes with high throughput in multi-hop wireless networks High-Throughput Metrics (HTMs).

Researchers prefer on-demand routing protocols in wireless ad-hoc networks, because they save resources such as bandwidth and processing power [23, 57]. To use HTMs in on-demand routing protocols, typically metrics of all links on a path need to be collected [31, 34]. Thus, protocols that use source routing are suitable to support HTMs, since they contain individual link information in their route-discovery messages. As a result, DSR (dynamic source routing) [58] is a natural choice when supporting an HTM. Both ETX and ETT have been implemented as a metric in DSR [31, 34]. However, the existing design of supporting HTMs in DSR is not efficient, because it collects link state information and runs Dijkstra's shortest path algorithm.

AODV (ad hoc on-demand distance vector) [85] is another well-known and well-studied on-demand routing protocol. However, when trying to support an HTM on AODV, we face problems because control messages in AODV typically do not contain individual link information. ETX is also implemented as a metric in DSDV (destination-sequenced distance-vector routing [84]) in [31]. However, DSDV works proactively instead of reactively. AODV-ST (spanning tree) [90] supports the ETT metric, but it is based on infrastructured wireless networks (we discuss the details in Section 7.3.2).

Therefore, the motivation of this work is to design an efficient and generalized scheme to support HTMs in on-demand routing protocols, including AODV. The main contributions of this work are as follows:

(i) We give a taxonomy of existing HTMs and propose an efficient and generalized approach called APM (accumulated path metric) to support HTMs in on-demand routing protocols. One advantage of APM is that it discovers the shortest path, in terms of an HTM, without collecting the network topology and without using a shortest path algorithm such as Dijkstra's algorithm.

(ii) We prove the correctness of APM, i.e., a route discovered by using APM is the shortest path. The proof ensures the applicability of APM in on-demand routing protocols and also indicates that the existing design of supporting HTMs in DSR can be considerably simplified.

(iii) We address the overhead of duplicate RREQ transmissions in supporting HTMs in on-demand routing protocols. We propose a *broadcast ordering* (BO) technique to suppress unnecessary RREQ transmissions.

(iv) We evaluate the performance of APM and BO in both AODV and DSR through simulations. The results verify the effectiveness of the proposed approaches. When analyzing the improvement achieved by the BO technique, we use least-squares polynomial fitting [27] to fit the data for the overhead of RREQ transmissions. This is a novel method for studying the performance of protocols designed for ad-hoc networks.

The rest of this chapter is organized as follows. In the next section, we discuss existing HTMs. The APM approach is presented in Section 7.3, and the broadcast ordering technique is presented in Section 7.4. Section 7.5 presents simulation setup and simulation results. We classify this chapter in Section 7.6.

7.2 High-Throughput Metrics (HTMs)

Researchers have proposed a number of HTMs to measure the qualities of wireless links and wireless paths composed of multiple hops. We can classify these metrics according to different criteria, reflecting their fundamental design and implementation choices. In this section, we give a taxonomy of existing HTMs and discuss the current approach that supports HTMs in DSR.

7.2.1 A Taxonomy of HTMs

We classify high-throughput routing metrics based on a taxonomy that consists of three categories: link quality measurement, path quality measurement, and radio model.

Link Quality Measurement

A fundamental issue in HTM design is the quality measurement of a wireless link. Existing schemes use a variety of methods to measure link qualities. We classify these methods into the following three categories:

(i) *Counting packets*. A node counts the number of packets received from each neighboring node. Packets that a node counts are either periodic probe packets specifically designed to test link quality, or data packets generated by applications and control packets generated by the routing protocol. Metrics in this category include expected transmission count (ETX) [31], expected transmission time (ETT) [34], and estimated delivery rate (or loss rate) [105]. We classify ETT in this category because it uses ETX.

(ii) *Measuring time*. A node measures the time needed to transmit a packet to a neighboring node. For example, the RTT (per-hop Round Trip Time) metric [9] is based on the round trip delay seen by unicast probes between neighboring nodes.

(iii) *Measuring Distance*. In geographic routing, a node may measure the distance to a neighbor and the cost (e.g., power consumption) to transmit a packet to that neighbor. The goal is to obtain a high advance-to-cost ratio. An example of such metrics is normalized advance (NADV) [66].

Path Quality Measurement

Path quality measurement refers to the operation of accumulating individual link metrics on a path to obtain an end-to-end metric for that path. Path quality measurement is associated with the method for the link quality measurement. The following operations are used to calculate a path metric:

(i) *Addition*. The path metric is the sum of metrics of each hop. The ETX scheme is a representative of this type.

(ii) *Multiplication*. The path metric is the product of metrics of each hop. An example is the delivery rate metric [105].

(iii) *Mixed operations*. The calculation of path metric involves mixed operations. For example, the ETT scheme applies a tunable weight to reflect the expected transmission time and the channel diversity. In other words, both addition and multiplication are used.

Radio Model

We use this criterion to classify the radio model that an HTM scheme is based on. In existing HTM schemes, two radio models are used: single channel and multiple channels.

(i) *Single channel*. Nodes in the network are equipped with homogeneous radio cards, and the cards operate on the same channel. The ETX scheme is based on this model.

(ii) *Multiple channels*. Nodes are equipped with heterogeneous radio cards (i.e., different standards such as 802.11a and 802.11b), or equipped with homogeneous radio cards, but the cards operate on multiple channels. The ETT scheme is proposed to work in the multiple-channel situation.

7.2.2 The Existing Approach to Supporting HTMs in DSR

DSR (dynamic source routing) [58] is a natural choice in supporting HTMs in on-demand routing protocols (e.g., [31, 34, 33]), because an HTM scheme typically needs to collect link state information to calculate the shortest path. In DSR, a source node explicitly designates the route to a destination in the header of data packets. Routes are established by recording traversed paths in RREQ (route request) messages and returning discovered paths in RREP (route reply) messages. In general, DSR is modified in the following ways to support an HTM.

First, when a node forwards a RREQ message, it appends not only its own address, but also the metric of the link (over which it received the RREQ) to the message.

Second, when a node receives a duplicate RREQ packet, instead of simply discarding it, the node broadcasts the RREQ if the accumulated metric in the message is better than the best metric it has forwarded for this route request. This approach increases the overhead of a route discovery, but it is necessary to collect link state information for the shortest path calculation. We discuss this issue in detail in Section 7.4 when presenting the broadcast ordering technique.

Third, the source node does not use the routes returned by RREP messages immediately. Instead, it stores the received routes in a *link cache*, where routes are saved as separate links (hops). Then the source runs Dijkstra's shortest-path algorithm on the link cache to find the shortest path to the destination.

We refer to this modified DSR as DSR-HTM in our discussion.

7.3 APM: Supporting HTMs in On-Demand Routing Protocols

In this section, we first discuss the motivation and then present the operation of the new APM approach. We prove the correctness of APM in Section 7.3.3, i.e., routes discovered by APM are the shortest paths. We discuss the issue of applying APM in more scenarios in Section 7.3.4: for example, in scenarios where the path metric is calculated using multiplication and where nodes communicate over multiple channels.

7.3.1 Motivations

The DSR-HTM approach discussed in Section 7.2.2 can find routes with high throughput. However, its operation is inefficient in two aspects: collecting link state information and running Dijkstra's shortest-path algorithm. These operations are expensive in terms of communication and processing overhead. Thus, our first motivation is to make the DSR-HTM approach more efficient.

The second motivation comes from the attempt to use HTMs in AODV (ad hoc on-demand distance vector) [85]. When supporting an HTM, AODV has difficulties because its route discovery messages do not contain individual link information. Thus, it is not able to collect link state information and run Dijkstra's shortest path algorithm. A possible solution is to let RREQ and RREP messages in AODV carry the entire path. However, this approach is not desirable because it deprives AODV of its simplicity. For example, RREQ and RREP messages would no longer be small and constant in size.

The above discussion motivates us to design an efficient and generalized approach to supporting HTMs in on-demand routing protocols. From the route discovery process of DSR-HTM, we make two observations:

First, intermediate nodes propagate not only the first received RREQ packet, but also duplicate RREQ packets that have a new better (shorter) path.

Second, to determine if a path is shorter than another, only the accumulated path metric, instead of individual link metrics, is used for comparison.

Therefore, our intuition is that among the RREQ packets received by the destination, the one with the best accumulated metric is likely to be the packet that traversed the shortest path from the source to the destination. In other words, it may not be necessary for the source to collect link-state information and run Dijkstra's shortest path algorithm.

Based on the observations and the intuition, we propose the APM approach, whose operation is presented next.

7.3.2 The APM Approach

APM (accumulated path metric) is an efficient and generalized approach to supporting HTMs in on-demand routing protocols for multi-hop wireless networks. We first use an ETX-like HTM scheme as an example when presenting APM. An ETX-like HTM scheme features the following: first, the metric of a path is the sum of link metrics on the path; second, radios on nodes work on the same channel. We discuss the operation of APM in other schemes, such as schemes that use multiplication in calculating path metric or use multiple channels, in Section 7.3.4.

In the following discussion, we denote a metric (M) in three contexts:

$M_{link}(u,v)$:	The metric of a link between nodes u and v.
$M_{path}(s,u)$:	The metric of a path from node s to node u. It is the accumulated value of M_{link}s on the path.
$M_{route}(s,d)$:	The metric of the route from source s to destination d. It is the M_{path} of the shortest path, which has the lowest M_{path} and is selected as the route.

In on-demand routing protocols, a source node discovers a route typically by flooding the entire or a part of the network with a RREQ message. Each RREQ message is assigned a sequence number ($broadcast\_id$). The combination of source address and broadcast id uniquely identifies a RREQ message in the network.

To support an HTM, we add a new field M_{path} in RREQ messages. The M_{path} field stores the accumulated metric for the path that a RREQ packet has traversed. When a source node s initiates a route discovery searching for a destination node d, the source sets the M_{path} in the RREQ message to zero.

Upon receiving a RREQ packet from a neighboring node u, an intermediate node v first updates the M_{path} in the packet as follows:

$$M_{path}(s,v) \leftarrow M_{path}(s,u) + M_{link}(u,v) \tag{7.1}$$

Node v forwards the RREQ packet if either of the following two conditions is satisfied: first, the RREQ is received for the first time; second, a duplicate RREQ packet is received, and the M_{path} of the packet is better than the best value that the node has seen so far for this route discovery. To determine whether a RREQ packet is new or duplicate, or is better or worse, each node maintains a ***RREQ_Seen*** table. Each entry in the table represents a route request that the node has seen. An entry is identified by the combination of source address and broadcast id. We define a field called $best\_path\_metric$ in the ***RREQ_Seen*** table. This field records the best M_{path} that the node has forwarded for a route discovery.

Routing protocols such as AODV maintain a reverse route to the source at intermediate nodes when they propagate a RREQ message. In this case, if node v decides to forward a RREQ, it updates the reverse route to source s. More specifically, node v sets node u, from which v receives the RREQ, as the next hop to source s.

Intermediate nodes do not reply to a RREQ, even if they have a route to the destination in their routing tables or route caches, because routes reported by intermediate nodes cannot ensure the best path metric.

When the destination receives the first RREQ packet, it does not send a RREP to the source immediately. Instead, it waits for some time (RREQ_WAIT_TIME) by using a timer. During this period, the destination may receive duplicate RREQ packets for this route discovery, which may contain better M_{path} values. When the timer expires, the destination selects the path with the best M_{path} value as the shortest path (the route) from the source, and sends a RREP message to the source. The source will subsequently use the received route to transmit data packets.

Comparing APM and DSR-HTM

Compared to the route discovery process in DSR-HTM, APM has the following differences: (i) A RREQ message in APM contains an accumulated metric for the path, instead of individual metrics for each link. Carrying only path metric means the byte overhead caused by using an HTM scheme is constant and small, which is critical in controlling the routing overhead. (ii) In APM, the route is selected by the destination. For a route discovery, the destination typically sends one RREP message to the source. In DSR-HTM, on the other hand, the destination sends a RREP message to the source for every received RREQ packet; the source stores the received paths in a link cache, and then runs Dijkstra's shortest path algorithm to calculate the route. The APM approach is more efficient in terms of communication and processing costs, since it does not collect link state information and does not run Dijkstra's shortest path algorithm.

Comparing APM and AODV-ST

Ramachandran et al. [90] propose AODV-ST (spanning tree) for routing in infrastructured mesh networks. AODV-ST proactively establishes spanning trees in mesh clusters. Our work differs from AODV-ST in the following ways. First, APM is a generalized approach for on-demand routing protocols. Second, we prove the correctness of APM (Section 7.3.3). Third, we study the difference between path metrics obtained by APM and theoretical metrics calculated by using Dijkstra's shortest path algorithm. The algorithm ran over the topology of the entire network. Finally, our broadcast ordering technique (Section 7.4) is complementary to AODV-ST.

7.3.3 Correctness

We now prove the correctness of APM, i.e., show that the route discovered by APM is the shortest path from the source to the destination. We make the following assumptions for the proof: the network is connected, no RREQ packet is lost, and when the destination selects a route, the propagation of all RREQ packets for that route discovery has finished. We discuss the impact of RREQ packet losses after the proof.

Definition 1 (Predecessor)**.** *For a route discovery, the predecessor of a node is the neighbor from which the node receives the RREQ with the best M_{path}.*

The role of a predecessor is specific for each route discovery, and the predecessor of the source is NULL. Given a network, we represent the topology as $G = (V, E)$, where V is the set of nodes (vertices) and E is the set of links. If we maintain a predecessor $\pi[v]$ for each node v, we obtain a *predecessor subgraph* $G_\pi = (V, E_\pi)$. In APM, after propagating all RREQ packets for a route discovery, a G_π is formed.

We need to prove that the path from the source node s to the destination node d in G_π is the shortest path between them in G.

Lemma 7.3.1. *In APM, for each route discovery, one RREQ packet is propagated through the shortest path from the source to the destination.*

Proof. The proof is by induction. Suppose the shortest path from s to d is: $s - v_1 - v_2 - ... - v_k - d$. If there exist multiple shortest paths from the source to the destination, we arbitrarily choose one.

Base Case: The first intermediate node v_1 forwards the RREQ message initiated by source s, because v_1 is a direct neighbor of s. After receiving the RREQ packet broadcast by s, v_1 forwards the packet either because it is the first message that it received for this route discovery or because it has the minimum M_{path} (zero). We assume that no RREQs are lost.

Inductive Hypothesis: Suppose the RREQ was forwarded sequentially by nodes s, v_1, ..., $v_i (i < k)$. Now we need to show that the RREQ will be forwarded by node v_{i+1}. Upon receiving the RREQ, node v_{i+1} updates the $M_{path}(s, v_{i+1})$ value of the packet according to Formula (7.1).

Assume to the contrary that the RREQ is not forwarded by node v_{i+1}. This means node v_{i+1} has received a RREQ with a smaller or equal $M'_{path}(s, v_{i+1})$ value, which represents the metric of another path $s - v_1^{'} - ... - v_j^{'} - v_{i+1}$. Then, using the “cut and paste” technique, we can have a new path from s to d: $s - v_1^{'} - ... - v_j^{'} - v_{i+1} - ... - d$.

If $M_{path}(s, v_{i+1}) > M'_{path}(s, v_{i+1})$, the new path is shorter than the original path, which contradicts our assumption that the original path is the shortest path. If $M_{path}(s, v_{i+1}) = M'_{path}(s, v_{i+1})$, then there exist multiple shortest paths from the source to the destination. In this case, we use the new path at the beginning of the proof.

Therefore, the RREQ is forwarded by all intermediate nodes along the shortest path: v_1, v_2, ..., v_k. After node v_k forwards the RREQ, destination d receives it. If there exist multiple shortest paths from the source to the destination, the path through which the RREQ reaches the destination first is selected by APM. This completes the proof. $\square$

The proof can also be understood in this way: subpaths of shortest paths are shortest paths. This property is shown in [29] (page 582, Lemma 24.1, optimal substructure of a shortest path). Thus, each intermediate node on the shortest path forwards the RREQ because the subpath is the shortest path from the source to the intermediate node, and the RREQ has the best M_{path} value.

Theorem 2. *APM finds the shortest path from the source to the destination.*

Proof. According to Lemma 7.3.1, one RREQ is propagated through the shortest path and reaches the destination. This RREQ has the best accumulated M_{path} value, and the path is selected by APM as the route for data packet transmission. If there exist multiple shortest paths, APM selects the path through which a RREQ reached the destination first. $\square$

Corollary 3. *APM builds a shortest-paths tree*[1]*, rooted at the source node.*

Proof. When we prove Lemma 7.3.1 and Theorem 2, the destination can be any node in the network, except for the source node. The shortest-paths tree is the predecessor subgraph G_π formed after the propagation of all RREQ packets for this route discovery. □

Impact of a RREQ packet loss: In a real network, RREQ packets may be lost due to collisions or timeouts. The loss may result in a route discovered by APM not the shortest. Instead, the route may be the second or third shortest path, for example. This issue is not a serious concern for the following reasons: (i) The major goal of using metrics other than hop count is to exclude bad-quality links from a route. A second-shortest path is acceptable and is sometimes unavoidable. (ii) As shown in our simulations in Section 7.5, APM finds the shortest path most of the time. (iii) The loss of RREQ packets exists in protocols such as DSR-HTM [31, 33] as well and affects routing performance similarly.

7.3.4 The Operation of APM in Other HTM Schemes

The APM approach discussed in Section 7.3.2 is based on an ETX-like scenario, where a path metric is the sum of link metrics on the path and radios work on a single channel. The APM approach can easily be applied to other scenarios. We now discuss the cases of other path metric operations (e.g., multiplication) and multiple channels, respectively.

Path Metric Operations Other Than Addition

Accumulating link metrics may use operations other than addition. For example, in the delivery-rate metric [105], the delivery rate of a path is the product of delivery rates of each link on the path. In this case, we can convert the delivery rate of a link to a log scale. Then, an addition in the log domain is equivalent to a multiplication of delivery rates. After converting multiplication operations to addition operations, Formula (7.1) can still be directly used. The conversion is also suggested in [105] to reduce the storage and the processing overhead.

If the accumulation involves mixed operations, then a RREQ message might need to carry more fields for calculating the path metric, and Formula (7.1) might need to be adjusted according to the definition of the specific HTM scheme being used.

[1]We use *paths* instead of *path* because we obtain paths from the source to all other nodes in the network.

Multiple-Channel Radios

To use APM in schemes where radios on nodes operate on multiple channels, a RREQ message may need to carry information for each channel. Using ETT as an example, APM needs a separate field to save the X value for each channel, where X means the sum of ETTs of the hops using the same channel [34]. Since the number of channels is limited, we expect a fixed-length RREQ message. To further reduce the size of RREQ messages, we can use a threshold to limit the number of candidate channels in a RREQ message, or use a filter to remove low-throughput channels from a RREQ message.

7.4 The Broadcast Ordering Technique

During a route discovery in APM and in schemes such as DSR-HTM, intermediate nodes broadcast not only the first received RREQ packet, but also duplicate RREQ packets that have better M_{path}s. This approach increases the overhead of route discovery, but it is necessary in order to find a route with the best metric.

In the original AODV and DSR, flooding is used for route discovery, where each node except the destination broadcasts the RREQ message at most once. The transmission overhead of flooding is $O(n)$, where n is the number of nodes in the network. In APM, because of duplicate RREQ transmissions, the overhead of route discovery is higher than just flooding. In the worst case, the overhead can be $O(n^2)$, because a node may receive and broadcast $O(n)$ RREQ packets, and there may be $O(n)$ downstream nodes.

The extra overhead incurred by duplicate RREQ transmissions is undesirable, because it may consume significant bandwidth and cause congestion. In order to reduce the overhead of duplicate RREQ transmissions, we propose a technique called *broadcast ordering* (BO) to schedule the transmissions of RREQ packets. The basic idea is to let RREQ packets with better M_{path}s propagate faster in the network and thus to likely suppress RREQ packets with worse M_{path}s.

7.4.1 The Operation of Broadcast Ordering

Before explaining broadcast ordering, it would be helpful to review the concept of broadcast jitter [103]. In a normal flooding operation, the source node originates a broadcast packet, and all direct neighbors receive the packet almost simultaneously. If hardware and load of nodes are similar, the neighbors handle and broadcast the packet at about the same time, which leads to collisions and packet losses. To solve this problem, a random amount of delay is introduced before a node hands over the packet from the network layer to the MAC layer. This time is referred to as *broadcast jitter*; it helps avoid all neighbors broadcasting simultaneously. The jitter time is uniformly distributed over a small interval, e.g., from 0 to 10 milliseconds.

The broadcast ordering schedules the transmission of RREQ packets in such a way that a packet from a link with better M_{link} is transmitted earlier, and the packet from a link with worse M_{link} is transmitted later. A delay called *broadcast ordering delay* is introduced for this purpose, which we denote as D_{BO}. D_{BO} determines the waiting time before a RREQ packet is handed over from the network layer to the MAC layer.

A D_{BO} is composed of two parts: a random broadcast jitter (D_{jitter}) as mentioned above and a delay proportional to the M_{link} of the link over which the current node received the RREQ packet. We call this delay *link metric delay* and denote it as D_{metric}.

To calculate D_{metric}, we map the metric of a link to a level value l, which is an integer from 0 to a maximum number L_{max}. In our simulations, L_{max} was 6. Then, the D_{metric} is calculated by multiplying l by a constant factor D_{const}, which was $10ms$ in our simulations. The D_{jitter} is calculated by multiplying D_{const} by a real number r uniformly selected from [0, 1). Thus, the total delay D_{BO} is calculated as follows:

$$\begin{aligned} D_{BO} &= D_{metric} + D_{jitter} \\ &= l * D_{const} + r * D_{const} \\ &= (l + r) * D_{const} \end{aligned} \tag{7.2}$$

For example, if a node receives a RREQ packet from a link with the metric level l equal to 2, and the values of D_{const} and r are $10ms$ and 0.48, respectively, D_{BO} would be $(2+0.48)*10ms = 24.8ms$.

Figure 7.1 shows an example of the operation of broadcast ordering. Node A broadcasts a RREQ packet, and nodes B, C, D, and E receive and broadcast it. Node F receives four copies and will broadcast some of them. Without BO, node F may broadcast the RREQ up to four times if packets with better metrics arrive later. Under BO, node F broadcasts the RREQ only once, because the packet from the best link (AC) is sent first, and packets received later are discarded since they have worse metrics.

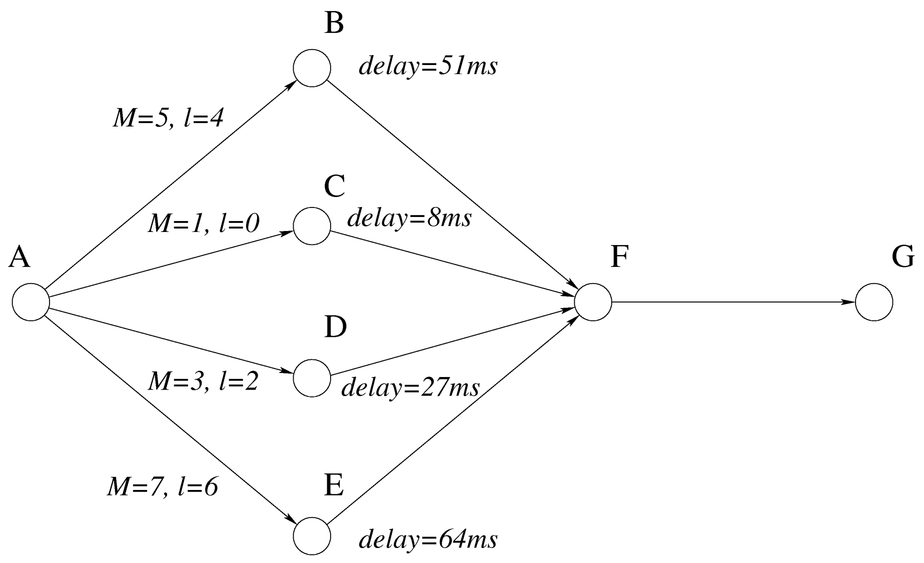


Figure 7.1: An example of the broadcast ordering. M refers to M_{link}, and *delay* refers to D_{BO}.

The broadcast ordering reduces the number of RREQ transmissions required by APM, and thus helps APM find the shortest path efficiently and effectively. Fewer transmissions mean less congestion, not to mention less battery drain. Less congestion increases the chance that a RREQ packet with the best M_{path} reaches the destination.

Ye et al. [106] propose a cost-field based approach to find minimum-cost paths from nodes to a sink in sensor networks. A backoff algorithm is used to reduce the broadcasts of advertisement messages. The backoff time is set to be proportional to the cost at a node. An important difference between the backoff algorithm and our broadcast ordering technique is that the backoff algorithm does not include a random broadcast jitter delay (i.e., D_{jitter} in Equation (7.2)). Thus, if two nodes calculate the same backoff time, they may broadcast at the same time and cause collision.

7.5 Performance Evaluation

We conducted simulations to study the performance of APM and the broadcast ordering technique. In this section, we first introduce the evaluation method and then present the results and analysis.

7.5.1 Evaluation Method

We conducted simulations using GloMoSim 2.03 [21]. The settings for radio model and MAC layer are the same as described in Chapter 3.4. The traffic was constant bit rate (CBR). CBR sources and destinations were randomly selected. Each flow did not change its source and destination for the lifetime of a simulation run. Each source originated four 512-byte data packets per second. The simulation results plotted in the graphs, except for the cumulative distribution function (CDF) graphs, were averaged over 20 simulation runs, each with a different seed. Error bars represent the 90% confidence interval for data points. We add a "goodness arrow" to each graph showing which direction is better. Each simulation lasted for 600 seconds. After a simulation began, each CBR source waited for a random time from 0 to 60 seconds before starting to send data packets, and ended sending data packets 20 seconds before the simulation finished.

We used a static network topology because HTM schemes are proposed for networks where nodes are typically not moving, such as WMNs (wireless mesh networks). As observed by Draves et al. [33], HTM schemes may not perform well in mobile scenarios, because they do not react quickly to topology changes. We used an ETX-like metric in simulations, where the metric of a path (M_{path}) is the sum of the metrics of each hop (M_{link}s). If a link exists between two nodes, meaning they are within each other's radio range, we assign a random metric to that link. The metric is an integer randomly chosen from 1 to 7. If no link, we set the metric between the two nodes to INFINITY (its value was 1000 in simulations).

We did not use a link probe process as in [31] and [34]. This omission would not compromise our evaluation for the following reasons. First, we focused on the ability of APM to discover shortest paths, instead of the assessment of link qualities. The random link metrics represent a general network scenario. Second, assessing link metrics does not necessarily require active link probing. Alternatively, passive methods, such as listening to the traffic, may be used to estimate the qualities of communication channels. The advantage of passive methods is to save channel bandwidth and battery, which are limited resources and are major constraints in wireless networks. The APM approach is independent of the underlying link metric assessment scheme.

The evaluation focused on two aspects. First, we studied the performance of APM by comparing practical and theoretical route metrics. A practical route metric is the metric of a route discovered by the routing protocol being used, and we denote it as M^a_{route}. A theoretical route metric is the metric of a route calculated by running Dijkstra's shortest path algorithm over the topology of the entire network. We denote it as M^d_{route}. Second, we studied the performance of the broadcast ordering technique. Particularly, we focused on the saving of route discovery overhead and the improvement of APM in finding shortest paths.

To compare the practical and the theoretical M_{route}s, when the source node of a route discovery received a route reply (RREP) message generated by the destination, we recorded the metric of the discovered route as M^a_{route}. Then, we ran Dijkstra's shortest path algorithm to calculate the shortest path between the source and the destination, and recorded the metric of the path as M^d_{route}. We compared these two values to study how close the M^a_{route} was to the M^d_{route}. Theoretically, they should be equal. However, due to collisions and losses of RREQ packets, M^a_{route} might be larger than M^d_{route}.

We simulated two scenarios by varying two network parameters. In the first scenario, the network size varied from 50 to 300 nodes and the number of CBR flows was one. The purpose of this study was to evaluate the effectiveness of APM and BO in different size networks. We used one CBR flow to eliminate collisions of RREQ packets from different flows. The network area was selected such that the node density was kept approximately constant, as shown in Table 7.1.

Table 7.1: Network sizes and areas.

Network Size	*Network Area (m^2)*
50	850×850
100	1200×1200
150	1470×1470
200	1700×1700
250	1900×1900
300	2080×2080

In the second scenario, the network had 100 nodes in an area of 1200×1200m^2, and the number of CBR flows was varied from 5 to 20. This study aimed to evaluate the performance of APM and BO when there were more flows in the network.

In each scenario, we simulated APM in both AODV and DSR routing protocols, which are denoted as AODV-APM and DSR-APM, respectively. We also simulated BO in both AODV-APM and DSR-APM. When BO is used, we denote them as AODV-APM-BO and DSR-APM-BO, respectively. In order to demonstrate the advantage of the proposed APM scheme over the existing approach to support HTMs in DSR (DSR-HTM, Section 7.2.2), we also simulated DSR-HTM. We will not present results for packet delivery ratio, control overhead, and end-to-end delay because they are not of primary interest to this chapter. These measurements are pertinent to the effectiveness of a particular HTM scheme. On the other hand, we focus on the capability of APM to find the shortest paths and the savings of control overhead by the BO technique.

7.5.2 Results and Analysis

Different Network Size

Figure 7.2 shows the M^a_{route}s of AODV-APM, AODV-APM-BO, DSR-APM, DSR-APM-BO, and DSR-HTM, respectively, with respect to the network size. The figure also shows the theoretical M^d_{route}, which was calculated by using Dijkstra's shortest path algorithm.

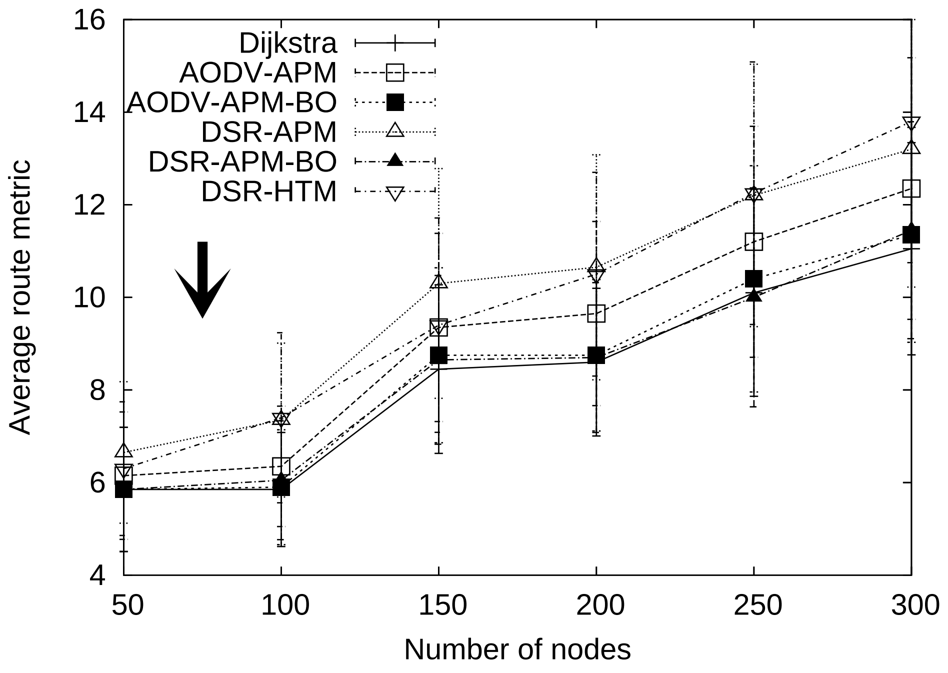


Figure 7.2: Different network size: average route metrics.

We observe that the M^a_{route} of AODV-APM is close to M^d_{route}. The difference between them becomes larger when the network size increases, but is still acceptable. For example, when the network size is 300, the M^a_{route} of AODV-APM is 12% larger than the M^d_{route}. DSR-APM performs worse than AODV-APM. For example, when the network size is 300, the M^a_{route} of DSR-APM is 19% larger than the M^d_{route}.

The difference between the M^a_{route}s and the M^d_{route} mainly comes from the collisions of RREQ packets. The transmission of a RREQ packet with a better M_{path} may collide with the transmissions of other packets and cannot be received by downstream nodes. The difference between the M^a_{route}s and the M^d_{route} increases as the network size increases, because larger networks experience more collisions. The collision is worse in DSR than in AODV because DSR records path in RREQ packets and results in larger RREQ packets. Larger packets require more transmission time and are more subject to interference.

Under BO, both AODV-APM-BO and DSR-APM-BO perform noticeably better than without BO. The M^a_{route}s of AODV-APM-BO and DSR-APM-BO are always very close to the M^d_{route}. Even when the network size is 300, the M^a_{route}s are only about 3% larger than the M^d_{route}.

Under BO, RREQ packets that have better metrics are given higher transmission priority, i.e., shorter delay. Thus, unnecessary transmissions for RREQ packets that have worse metrics are likely to be suppressed, reducing the overhead of route discovery, i.e., the number of RREQ transmissions. Therefore, collisions are mitigated, and a RREQ packet with the best metric is more likely to reach the destination.

In Figure 7.2, we observe that the performance of DSR-APM is very comparable to DSR-HTM. This result indicates that the APM approach accomplishes our initial goal: to find a more efficient way to support HTMs in DSR. DSR-APM performs as well as DSR-HTM, without collecting link state information and without running Dijkstra's shortest path algorithm. When using broadcast ordering, DSR-APM-BO performs significantly better than DSR-HTM.

Next, we present the results of the route discovery overhead (number of RREQ and RREP transmissions) in AODV-APM, DSR-APM, and DSR-HTM, and present the route discovery overhead saved by the broadcast ordering.

Route Discovery Overhead

Figure 7.3 shows the average numbers of RREQ transmissions per route discovery in AODV-APM, AODV-APM-BO, DSR-APM, DSR-APM-BO, and DSR-HTM, respectively. The "Flooding" curve represents the number of transmissions for new RREQ packets in AODV-APM. In other words, transmissions for duplicate RREQ packets are not counted. Thus, it is equivalent to the overhead of the flooding approach.

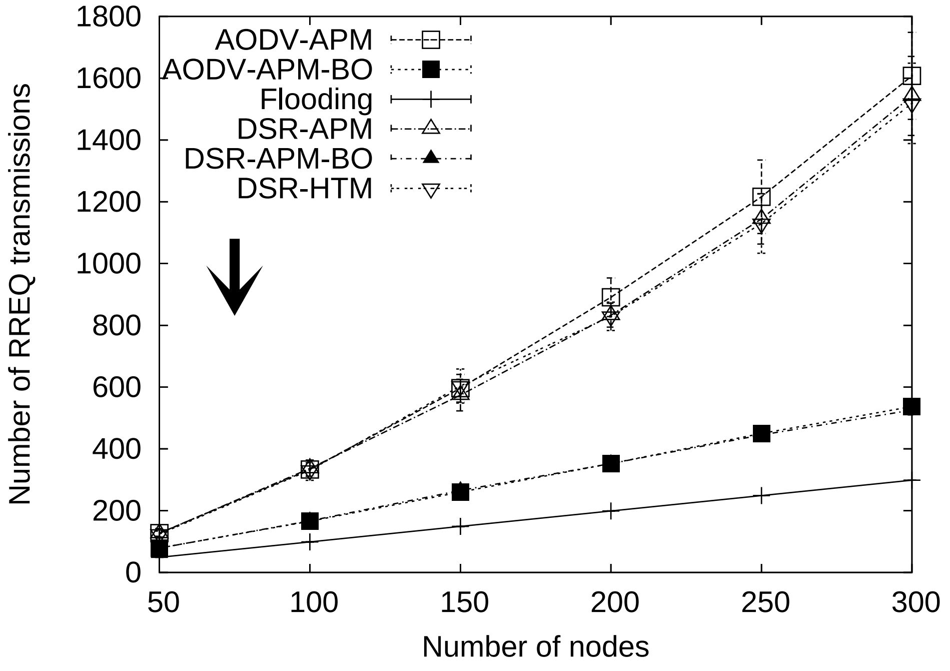


Figure 7.3: Different network size: RREQ transmissions.

We observe that AODV-APM, DSR-APM, and DSR-HTM incur about three times the overhead of Flooding. In order to analyze the relation between overhead ($F(n)$) and the network size (n), we used a second-degree polynomial function to fit the data for AODV-APM by least-squares fitting [27], resulting in:

$$F_{\text{AODV-APM}}(n) = 0.00841n^2 + 2.97n - 42.9, \tag{7.3}$$

where the standard deviation is 6.9 and the R-square (or the coefficient of determination, an indicator of how well the model fits the data) is 0.9999. The coefficients in the function indicate the overhead of RREQ transmissions. Larger coefficients mean larger overhead. We notice that the coefficient of the first-degree term is nearly 3, whereas the coefficient of the second-degree term is not large (0.00841). However, the second-degree term will exceed the first-degree term when n is large enough ($n > 352$ in Equation (7.3)). These results show that duplicate RREQ transmissions significantly increase the overhead of route discovery, and the problem should be properly taken care of.

With the broadcast ordering, we observe in Figure 7.3 that both AODV-APM-BO and DSR-APM-BO consistently reduce the overhead by more than half compared to AODV-APM and DSR-

APM, respectively. The curves of AODV-APM-BO and DSR-APM-BO are much closer to the curve of Flooding. We also computed a least-squares fit for AODV-APM-BO, resulting in:

$$F_{\text{AODV-APM-BO}}(n) = 0.000114n^2 + 1.81n - 13.5, \tag{7.4}$$

where the standard deviation is 3.4 and the R-square is 0.9998. Compared to Equation (7.3), the coefficient of the second-degree term is reduced by nearly two magnitudes and the coefficient of the first-degree term is reduced by nearly half. These results verify the effectiveness of the broadcast ordering technique.

Figure 7.4 shows the number of RREP transmissions per route discovery in AODV-APM, AODV-APM-BO, DSR-APM, DSR-APM-BO, and DSR-HTM, respectively. We observe that APM schemes (including AODV-APM, AODV-APM-BO, DSR-APM, and DSR-APM-BO) have a very low number of RREP transmissions, whereas the number of RREP transmissions in DSR-HTM is much higher. APM does not collect topology information in a route discovery process. Therefore, the destination sends only one route reply message to the source. On the other hand, DSR-HTM collects topology information at the source. Therefore, the destination replies every request it receives, and these replies cause a considerable control overhead.

Different Number of Flows

Figure 7.5 shows the M^a_{route} figures for AODV-APM, AODV-APM-BO, DSR-APM, DSR-APM-BO, and DSR-HTM, respectively, with respect to the number of flows. The figure also shows the M^d_{route} calculated by using Dijkstra's shortest algorithm. We observe that generally the M^a_{route} of AODV-APM is close to M^d_{route}. The difference between them becomes larger when the number of flows increases. When the number of flows is 5, AODV-APM is 11$\%$ larger. When the number of flows is 20, AODV-APM is 31$\%$ larger. The trend of the DSR-APM curve is similar to that of AODV-APM. However, DSR-APM consistently performs worse than AODV-APM. When the number of flows increases, the network becomes more congested, and more RREQ packets are lost due to collisions. As a result, the performance of APM is affected.

Under BO, AODV-APM-BO and DSR-APM-BO perform noticeably better than AODV-APM and DSR-APM, respectively. When the number of flows is not large (≤ 10 in the figure), the results of AODV-APM-BO and DSR-APM-BO are very close to the theoretical value. For example, the

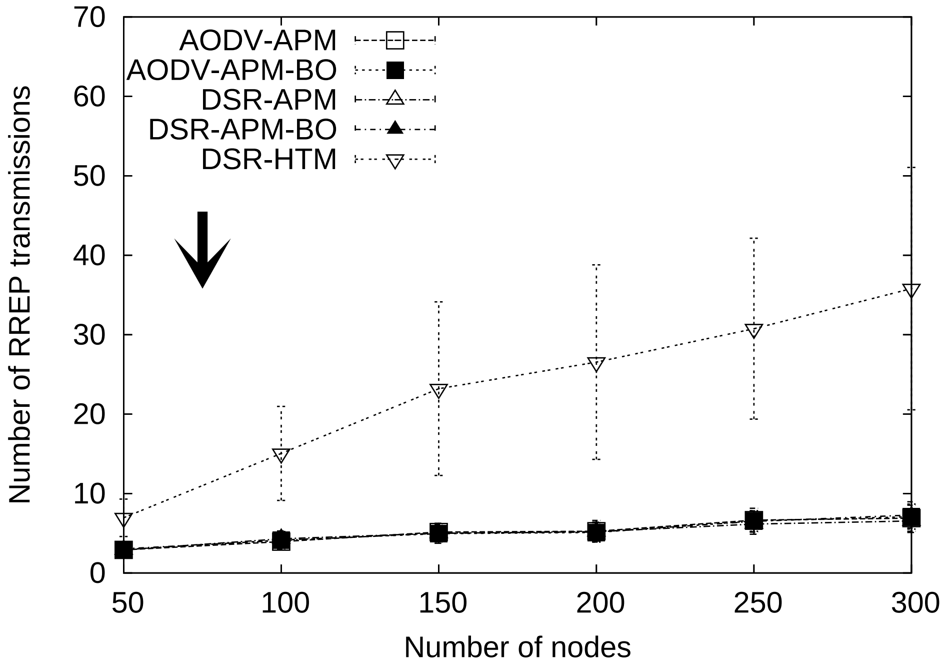


Figure 7.4: Different network size: RREP transmissions.

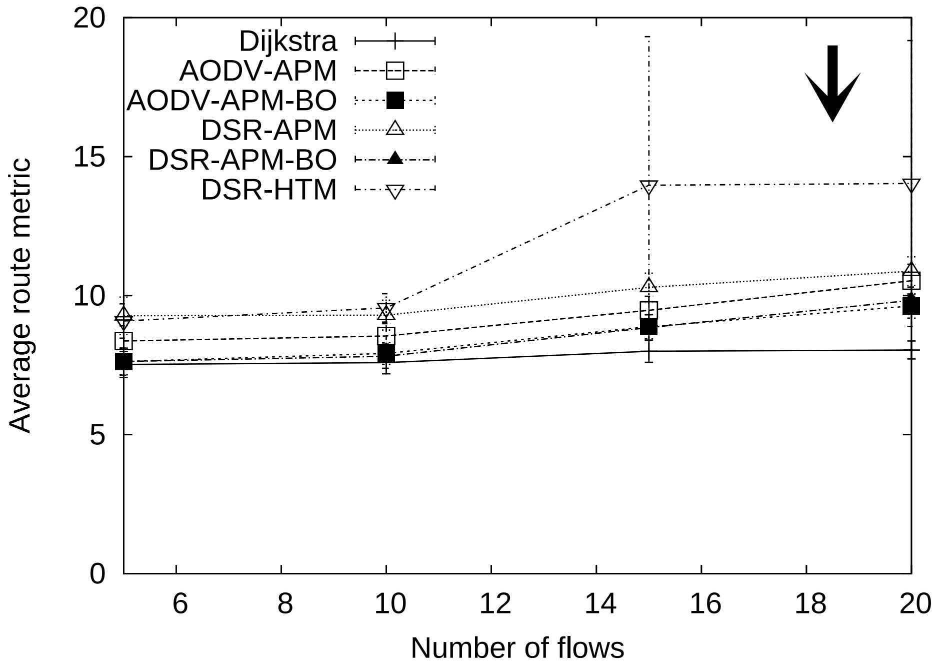


Figure 7.5: Different number of flows: average route metrics.

M^a_{route} of AODV-APM-BO is only 4% larger than the M^d_{route} when the number of flows is 10. The performance of AODV-APM-BO and DSR-APM-BO decrease when there are more flows, because of the collisions of RREQ packets.

All APM schemes perform considerably better than DSR-HTM. For example, when the number of flows is 15, the M^a_{route} of DSR-APM is 26% smaller than the M^a_{route} of DSR-HTM, and the M^a_{route} of DSR-APM-BO is 37% smaller than the M^a_{route} of DSR-HTM.

Figure 7.6 shows the cumulative distribution function (CDF) of the scaled difference d between the M^a_{route} of AODV-APM-BO and M^d_{route}, i.e., $d = (M^a_{route} - M^d_{route})/M^d_{route}$. Similar to the trend in Figure 7.5, when the number of flows increases, the CDF becomes worse. In other words, given an upper bound P, the probability of $d \leq P$ becomes lower. However, in most cases, d is not more than 1. For example, even when the number of flows is 20, the probability of $d \leq 1$ is 87%.

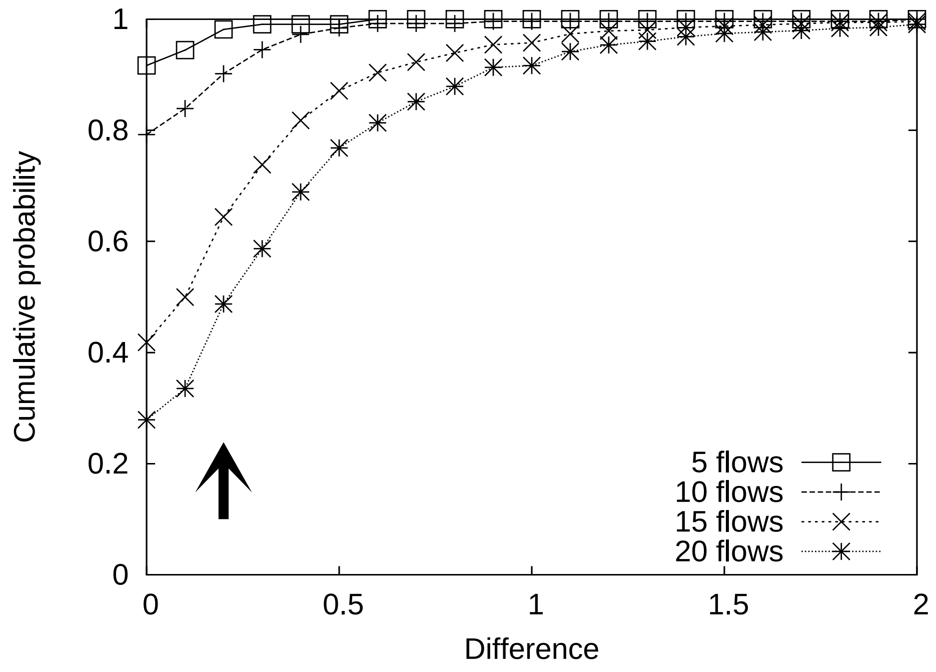


Figure 7.6: Different number of flows: CDF of scaled difference between the M^a_{route} of AODV-APM-BO and M^d_{route}.

In summary, the results in Sections 7.5.2 and 7.5.2 show that: (i) The APM approach works well in finding a route with the best metric. The M^a_{route}s discovered by AODV-APM and DSR-APM generally are close to the M^d_{route} obtained by running Dijkstra's centralized shortest path algorithm. (ii) The broadcast ordering technique significantly reduces the route discovery overhead by suppressing unnecessary RREQ transmissions. (iii) The performance of APM in AODV is better than in DSR because control packets in DSR are larger and are more subject to collisions. (iv) Compared to DSR-HTM, APM schemes, especially AODV-APM-BO and DSR-APM-BO, perform significantly better and operate much more efficiently.

7.6 Summary

Researchers have proposed high-throughput metrics (HTMs) to select high throughput routes in multi-hop wireless networks, and thus improve routing performance. The existing design to support HTMs in DSR is not efficient, because it collects link state information and runs Dijkstra's shortest path algorithm. AODV lacks support for HTMs because its route discovery messages do not contain individual link information.

We gave a taxonomy of existing HTM schemes and proposed an efficient and generalized approach called APM (accumulated path metric) to support HTMs in on-demand routing protocols. One advantage of APM is that it frees nodes from the burden of collecting the topology information and running Dijkstra's shortest path algorithm. In APM, a RREQ packet contains an accumulated metric for the path it traversed, instead of individual metrics for each link. The RREQ with the best accumulated metric received by the destination represents the shortest path. We proved the correctness of APM, i.e., the discovered route is the shortest path from the source to the destination.

Moreover, we addressed the issue of duplicate RREQ (route request) transmissions in existing HTM schemes. We proposed the broadcast ordering technique to suppress the transmissions of RREQ packets that have bad metrics.

We simulated APM in both AODV and DSR, with and without the broadcast ordering technique. Our simulation results show that the metric of a route discovered by APM is very close to the theoretical value calculated by Dijkstra's shortest path algorithm. Our results also show that the broadcast ordering technique significantly reduces the overhead of RREQ transmissions and improves the performance of APM.

Chapter 8

Conclusion

Mobile Ad-hoc Networks (MANETs) have received considerable research interest in recent years. Due to reasons such as node mobility and channel congestion, it is challenging to design routing protocols for MANETs. This dissertation has investigated a wide range of issues on routing protocols for MANETs. Specifically, we have focused on scalability, efficiency and path metrics. In this chapter, we conclude this dissertation by summarizing our contributions and suggesting directions for future research.

8.1 Contributions

Expecting future large wireless networks that operate in a peer-to-peer manner, we first focused on the scalability of routing protocols for MANETs. We have designed a new routing model called Way Point Routing (WPR). WPR selects a number of nodes on a route as waypoint nodes and divides the route into segments at the waypoints. Waypoint nodes run a high-level inter-segment routing protocol, and nodes on each segment run a low-level intra-segment routing protocol. When a node on a route moves away or fails, WPR finds a new path only for the broken segment. As an instantiation of WPR, we use DSR and AODV as the inter-segment and the intra-segment routing protocols, respectively. We term this instantiation as DSR Over AODV (DOA) routing protocol. We have also presented two novel techniques for DOA, an efficient loop detection method and a multi-target route discovery. The results of an extensive simulation study show that DOA scales well for large networks with more than 1000 nodes, incurring about 60%-80% less overhead than AODV, while its performance is better than or comparable to AODV and DSR with respect to other metrics .

The loss of route reply (RREP) messages can seriously impair the performance of on-demand routing protocols. We have proposed the idea of Salvaging Route Reply (SRR) to improve the performance of on-demand routing protocols. When an intermediate node cannot forward a RREP message to the intended next-hop node, the node becomes a salvor. We have presented two schemes to salvage an undeliverable RREP. In scheme one, a salvor actively sends a one-hop salvage request message to find an alternative path to the source. Most of the time, neighboring nodes are able to reply this message with another path because they just processed the route request (RREQ) message for the on-going route discovery. In scheme two, nodes in the network passively learn a backup path to the source from duplicate RREQ packets. A salvor uses the backup path to forward a RREP when it cannot deliver the RREP using the original path. Furthermore, we presented the design of two SRR schemes in AODV and proved that routes are loop-free after a salvage. We have conducted extensive simulations to evaluate the performance of SRR. The simulation results confirm the effectiveness of SRR.

Inspired by the idea of carpooling in real life, we have proposed Multiple-Target Route Discovery (MTRD) for on-demand routing protocols. MTRD aggregates multiple route requests into one RREQ message and discovers multiple targets simultaneously. MTRD is based on the observation that RREQ packets account for the dominant control overhead in on-demand routing protocols for mobile ad-hoc networks. Usually, a node relays a significant number of RREQ packets for other nodes. These RREQs provide opportunities of route discoveries for such relay nodes. In MTRD, when a source needs to perform a route discovery, instead of immediately broadcasting a new RREQ message, it first tries to attach its request to the existing RREQ packets that it relays. MTRD optimizes routing performance by reducing the number of regular route discoveries. The results of an extensive simulation study show that MTRD significantly improves routing performance.

We have observed that route discovery in MANETs operates in a source-initiated manner. This unilateral operation is unbalanced and is inherently not ideal for MANETs. We have developed a new scheme called Bilateral Route Discovery (BRD), where both source and destination actively participate in a route discovery process. BRD consists of two halves: a *source route discovery* (srd) and a *destination route discovery* (drd), each searching for the other. Nodes in the intersection of srd and drd learn paths to both source and destination, and these nodes can send RREP messages to the source of the route discovery. BRD has the potential to reduce the control overhead by one half. As an underlying protocol for BRD, we have developed Gratuitous Route Error Reporting (GRER). GRER bypasses the failed link and notifies the destination of a broken route. The destination can thus play an active role in the upcoming route re-discovery. Our extensive simulation study confirms the effectiveness of the BRD approach.

Finally, we notice that existing schemes to support High-Throughput Metrics (HTMs) are not efficient because they collect the topology information of a network and run a centralized shortest-path algorithm. We have proposed an efficient and generalized approach called Accumulated Path Metric (APM) for supporting HTMs in on-demand routing protocols. One advantage of APM is that it is able to find the shortest path, with respect to a particular metric, without collecting topology information and without running a shortest-path algorithm. We have proved the correctness of APM. Moreover, we have addressed the problem of duplicate RREQ transmissions with existing HTM schemes and have presented the Broadcast Ordering (BO) technique to suppress unnecessary RREQ transmissions. We have studied the performance of APM and BO in both AODV and DSR. The results verify the effectiveness of the proposed approaches.

8.2 Future Research Directions

Future research involves two aspects. The first aspect is to improve and extend our current work. Possible improvements include better schemes, theoretical insights, more comprehensive performance evaluations, and combining all the benefits to build a new suite of routing protocols. The second aspect is to explore other areas in wireless networking, such as MAC layer issues, location service, security, and different network types like mesh networks and personal area networks.

8.2.1 Improving and Extending Current Work

The WPR model is a promising framework for designing routing protocols for MANETs, especially for medium to large networks. WPR provides the following future research opportunities:

- **Waypoint selection.** Currently, when selecting waypoint nodes to divide a route into segments, we select nodes that divide the route into segments with roughly equal length. This method does not consider factors such as node speed. We plan to improve WPR by making a better selection of waypoint nodes. For example, selecting nodes with lower speed as waypoint nodes may lead to a more stable route.

- **Common segment.** In the current implementation of WPR/DOA, selecting waypoint nodes for different routes works independently. If different routes partially share a common path, selecting common waypoint nodes will result in common segments for these routes. Then, when a common segment is broken, WPR only needs to conduct the intra-segment route repair once. This approach is likely to further reduce the overhead of route maintenance but may increase congestion.

- **System parameters.** Several system parameters are important for WPR and may affect routing performance noticeably. These parameters include: default segment length, maximum segment length, and maximum number of waypoints. We plan to study the impacts of these parameters and adjust them using adaptive mechanisms.

- **Testing other protocols.** Currently we are using DSR over AODV (DOA) as an implementation of WPR. Theoretically, WPR can use many existing routing protocols. Depending on particular applications, protocols other than DSR and AODV may be more suitable. For example, when nodes are equipped with a GPS device, location-based routing protocols work better than topology-based routing protocols [43, 102]. We plan to study these scenarios and the suitable inter-segment and intra-segment routing protocols for WPR.

- **Security.** When comparing DSR to AODV, we notice that DSR is easier to secure than AODV. In DSR, a source node designates the path to route data packets. Therefore, DSR can select a route consisting of nodes that the source trusts. A considerable amount of research on security is based on DSR [22, 55, 71, 76]. Hu [54] and Gupte [48] present surveys on security issues in MANETs. We choose DSR as the high-level inter-segment routing protocol partially because of DSR's easy-to-secure characteristic. We plan to study possible attacks to WPR/DOA and develop proper schemes to secure WPR/DOA.

SRR focuses on the loss of RREP messages. Our future research on SRR includes these directions: studying the performance of SRR on other major routing protocols such as DSR, and theoretically analyzing the overhead saved by SRR. The differences between AODV and DSR may pose challenges to the design of SRR. For example, during a route discovery in AODV, the destination only answers the first received RREQ packet. However, in DSR, the destination answers every received RREQ packet. Thus, when applied to DSR, SRR does not need to salvage all the RREPs initiated by the destination. In addition to a simulation study, we plan to theoretically analyze the performance of SRR. The analysis may focus on these aspects: the expected number of RREQ transmissions for a route discovery, the probability of successfully salvaging a RREP, and the expected number of RREQ transmissions saved by SRR.

MTRD improves routing performance by applying the idea of carpooling to MANETs. Our future research on MTRD includes the following directions. First, we are interested in evaluating MTRD in other competitive on-demand routing protocols. Second, we plan to let nodes activate MTRD adaptively. For example, a node enables MTRD when the number of RREQs that it relays exceeds a certain rate. We will also try to use some heuristics in MTRD. For example, a node does not attach its request to every RREQ coming from the same neighbor. Third, we hope to theoretically analyze the performance of MTRD. We are interested in questions such as the probability that an attached request reaches its target and the overhead comparison between MTRD and regular route discovery. Moreover, we are interested in applying the idea of carpooling to other packets in MANETs, such as data packets.

BRD allows both source and destination to participate in a route discovery process. Currently, the success ratio of GRER is not very high. In the future, we hope to improve the success ratio of GRER. We will investigate other approaches to signaling the destination node of a broken route of the route error message. We may determine whether a link is broken using a timer or using a separate narrow-bandwidth but long-range control channel to notify a destination node directly.

APM provides a generalized approach to supporting HTMs. In the future, we plan to test other metrics in addition to ETX. We are also interested in studying the performance of APM in other on-demand routing protocols. Another important direction is to measure link quality using passive methods. Existing work assesses link metric actively using link probe messages. These messages consume resources such as bandwidth and energy considerably. We are interested in designing passive methods, such as listening to the traffic, to estimate the quality of communication channels.

8.2.2 Other Research Directions

As a new type of network, MANETs are promising but far from mature, not only in terms of application, but also in terms of technology. Even if there are few widespread applications for MANETs, we can view MANETs as the prototype of a number of other types of wireless networks that operate in an ad-hoc manner. New findings and improvements on MANETs will potentially benefit all ad-hoc wireless networks. If we can identify and address some fundamental problems in MANETs, we will be able to gain insights into other ad-hoc wireless networks and improve their applicability and practicability.

By adding or removing certain constraints such as mobility and power supply, we can transform a MANET into different variations. For example, nodes in a wireless mesh network (WMN) typically do not move and are not powered by battery; bluetooth uses short-range radio to establish personal area networks (PANs); and disruption-tolerant networking (DTN) addresses technical issues in extreme environments that lack continuous network connectivity. These networking technologies have received considerable interest from both academia and industry.

In the future, in addition to the area of routing protocols, we will extend our research interests to other challenges in wireless ad-hoc networking. Such challenges include designing medium access schemes that are suitable for wireless networks that feature a peer-to-peer operation, saving energy in power-constrained environments, network automatic configuration, enhancing network security and node cooperation to address billing issues.

We will be interested in not only infrastructure-less but also infrastructured wireless networks, such as cellular networks and wireless LANs. We will develop schemes that integrate the ad-hoc networking technologies to traditional infrastructured wireless networks. This integration is likely to improve the service quality and the network coverage. Moreover, in case that the fixed infrastructure is not functional (e.g., in a disaster scenario), the ad-hoc networking capability may provide some basic network connections.

A technology will not become widespread if it cannot find some popular applications. We feel compelled to apply techniques in ad-hoc networking to the domain of civil applications. We will focus on applying ad-hoc networking technology to various fields such as home automation and intra/inter-vehicle communication.

Bibliography

[1] Internet Usage Statistics, http://www.internetworldstats.com/stats.htm.

[2] Roofnet Project, http://pdos.csail.mit.edu/roofnet/.

[3] NetEquality, http://www.netequality.org/.

[4] Interplanetary Internet Project, http://www.ipnsig.org/.

[5] ExScal Project, http://www.cast.cse.ohio-state.edu/exscal/.

[6] Terminodes Project, http://www.terminodes.org/.

[7] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Standards Department, IEEE Standard 802.11-1997, 1994.

[8] IEEE 802.11. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Aug. 1999.

[9] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou. A multi-radio unification protocol for ieee 802.11 wireless networks. In *BROADNETS '04*, pages 344–354, Washington, DC, USA, 2004. IEEE Computer Society.

[10] S. Agarwal, A. Ahuja, J. Singh, and R. Shorey. Route-lifetime assessment based routing (rabr) protocol for mobilead-hoc networks. In *IEEE ICC*, June 2000.

[11] I. F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: a survey. *Comput. Netw. ISDN Syst.*, 47(4):445–487, 2005.

[12] B. Awerbuch, D. Holmer, and H. Rubens. High throughput route selection in mult-rate ad hoc wireless networks. Technical report, Johns Hopkins CS Dept, March 2003. v 2.

[13] R. Bagrodia, R. Meyer, M. Takai, Y. an Chen, X. Zeng, J. Martin, and H. Y. Song. Parsec: A parallel simulation environment for complex systems. *Computer*, 31(10):77–85, 1998.

[14] R. Bai and M. Singhal. Salvaging route reply for on-demand routing protocols in mobile ad-hoc networks. In *MSWiM 2005: Proceedings of The 8th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, oct 2005.

[15] R. Bai and M. Singhal. DOA: DSR over AODV routing for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(10):1403–1416, 2006.

[16] R. Bai and M. Singhal. BRD: Bilateral route discovery in mobile ad hoc networks. In *IFIP Networking 2007: the sixth International Conferences on Networking*, Atlanta, Georgia, May 2007.

[17] R. Bai and M. Singhal. Carpooling in mobile ad hoc networks: the case of multiple-target route discovery. In *WiOpt 2007: 5th Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, Limassol, Cyprus, April 2007.

[18] R. Bai and M. Singhal. Route discovery in mobile ad hoc networks: From unilaterality to bilaterality. In *Mobiquitous 2007: the 4th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, Philadelphia, PA, August 2007.

[19] R. Bai and M. Singhal. Enhancing performance by salvaging route reply messages in on-demand routing protocols for manets. *Ad Hoc & Sensor Wireless Networks*, accepted, 2007.

[20] R. Bai, M. Singhal, and Y. Wang. On supporting high-throughput routing metrics in on-demand routing protocols for multi-hop wireless networks. *J. Parallel Distrib. Comput.*, 67(10):1108–1118, 2007.

[21] L. Bajaj, M. Takai, R. Ahuja, R. Bagrodia, and M. Gerla. Glomosim: A scalable network simulation environment. Technical Report 990027, 13, 1999.

[22] R. B. Bobba, L. Eschenauer, V. Gligor, and W. Arbaugh. Bootstrapping security associations for routing in mobile ad-hoc networks. Technical report, Institute for Systems Research, UMd, May 2002.

[23] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *MobiCom*, pages 85–97, 1998.

[24] R. Castaneda and S. R. Das. Query localization techniques for on-demand routing protocols in ad hoc networks. In *MobiCom '99*, pages 186–194, New York, NY, USA, 1999. ACM Press.

[25] N. Chang and M. Liu. Revisiting the TTL-based controlled flooding search: Optimality and randomization. In *MobiCom '04*, pages 85–99. ACM Press, 2004.

[26] T.-W. Chen and M. Gerla. Global state routing: A new routing scheme for ad-hoc wireless networks. In *IEEE International Communications Conference, ICC98*, pages 171–175, Atlanta, GA, USA, June 1998. IEEE, IEEE.

[27] W. Cheney and D. Kincaid. *Numerical Mathematics and Computing, 4th edition*. International Thomson Publishing, 1998.

[28] C.-C. Chiang, H.-K. Wu, W. Liu, and M. Gerla. Routing in clustered multihop mobile wireless networks with fading channel. In *SICON'97*, pages 197–211, Singapore, April 1997.

[29] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, September 2001.

[30] L. H. M. K. Costa, M. D. D. Amorim, and S. Fdida. Reducing latency and overhead of route repair with controlled flooding. *Wirel. Netw.*, 10, 2004.

[31] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *MobiCom '03*, pages 134–146, New York, NY, USA, 2003. ACM Press.

[32] S. R. Das, C. E. Perkins, and E. E. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *INFOCOM*, pages 3–12, 2000.

[33] R. Draves, J. Padhye, and B. Zill. Comparison of routing metrics for static multi-hop wireless networks. In *SIGCOMM '04*, pages 133–144, New York, NY, USA, 2004. ACM Press.

[34] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *MobiCom '04*, pages 114–128, New York, NY, USA, 2004. ACM Press.

[35] R. Dube, C. D. Rais, K.-Y. Wang, and S. K. Tripathi. Signal stability-based adaptive routing (SSA) for ad hoc mobile networks. In *IEEE Personal Communications Magazine*, pages 36–45. IEEE, February 1997.

[36] E. Duros, W. Dabbous, H. Izumiyama, N. Fujii, and Y. Zhang. A link-layer tunneling mechanism for unidirectional links. RFC 3077, http://www.ietf.org/rfc/rfc3077.txt, Mar. 2001.

[37] J. Eriksson, M. Faloutsos, and S. Krishnamurthy. Scalable ad hoc routing: The case for dynamic addressing. In *INFOCOM'04*, Hong Kong, China, Mar 2004.

[38] K. Fall and K. Varadhan. The ns Manual. http://www.isi.edu/nsnam/ns/nsdocumentation.html.

[39] B. Fenner. Experimental values in IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP headers. RFC 4727, http://www.ietf.org/rfc/rfc4727.txt, November 2006.

[40] E. Gafni and D. Bertsekas. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Transactions on Communications*, 29(1):11–18, Jan 1981.

[41] J. J. Garcia-Luna-Aceves and M. Spohn. Source-tree routing in wireless networks. In *ICNP '99: Proceedings of the Seventh Annual International Conference on Network Protocols*, page 273, Washington, DC, USA, 1999. IEEE Computer Society.

[42] M. Gerharz, C. de Waal, M. Frank, and P. Martini. Link stability in mobile wireless ad hoc networks. In *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN'02)*, pages 30–39, Tampa, FL, November 2002.

[43] V. C. Giruka and M. Singhal. Two scalable location service protocols for wireless ad-hoc networks. *Pervasive and Mobile Computing (PMC)*, 2(3):262–285, 2006.

[44] V. C. Giruka, M. Singhal, and S. P. Yarravarapu. A path compression technique for on-demand ad-hoc routing protocols. In *IEEE MASS '04*.

[45] T. Goff, N. Abu-Ghazaleh, D. Phatak, and R. Kahvecioglu. Preemptive routing in ad hoc networks. *Journal of Parallel and Distributed Computing*, 63:123–140, 2003.

[46] T. Goff, N. B. Abu-Ghazaleh, D. S. Phatak, and R. Kahvecioglu. Preemptive routing in ad hoc networks. In *MobiCom '01*, pages 43–52, New York, NY, USA, 2001. ACM Press.

[47] C. Gui and P. Mohapatra. Short: Self-healing and optimizing routing techniques for mobile ad hoc networks. In *MobiHoc '03*, pages 279–290, New York, NY, USA, 2003. ACM Press.

[48] S. Gupte and M. Singhal. Secure routing in mobile wireless ad hoc networks. *Ad Hoc Networks*, pages 151–174, August 2003.

[49] Z. J. Haas and M. R. Pearlman. The zone routing protocol (ZRP) for ad-hoc networks. IETF MANET working group, Internet Draft, June 1999.

[50] Y. Han, R. J. La, and H. Zhang. Path selection in mobile ad-hoc networks and distribution of path duration. In *IEEE Infocom*, 2006.

[51] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang. A group mobility model for ad hoc wireless networks. In *MSWiM '99*, pages 53–60, New York, NY, USA, 1999. ACM Press.

[52] Y.-C. Hu and D. B. Johnson. Caching strategies in on-demand routing protocols for wireless ad hoc networks. In *MobiCom '00*, pages 231–242. ACM Press, 2000.

[53] Y.-C. Hu and D. B. Johnson. Design and demonstration of live audio and video over multi-hop wireless networks. In *MILCOM*, 2002.

[54] Y.-C. Hu and A. Perrig. A survey of secure wireless ad hoc routing. *IEEE Security and Privacy*, 2(3):28–39, 2004.

[55] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: a secure on-demand routing protocol for ad hoc networks. In *MobiCom '02*, pages 12–23, New York, NY, USA, 2002. ACM Press.

[56] P. Jacquet, P. Mhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol. In *IEEE INMIC'01*, Lahore, Pakistan, December 2001.

[57] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario-based performance analysis of routing protocols for mobile ad-hoc networks. In *MobiCom '99*, pages 195–206, New York, NY, USA, 1999.

[58] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[59] D. B. Johnson, D. A. Maltz, and Y.-C. Hu. The dynamic source routing protocol for mobile ad hoc networks (dsr). InternetDraft, http://tools.ietf.org/html/draft-ietf-manet-dsr-10.

[60] D. B. Johnson, D. A. Maltz, and Y.-C. Hu. The dynamic source routing protocol for mobile ad hoc networks (DSR). InternetDraft, http://tools.ietf.org/html/draft-ietf-manet-dsr-09, Apr. 2003.

[61] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *MobiCom '00*, pages 243–254, New York, NY, USA, 2000. ACM Press.

[62] Y. Kim, S. Choi, K. Jang, and H. Hwang. Throughput enhancement of ieee 802.11 wlan via frame aggregation. In *IEEE VTC'04-Fall*, Los Angeles, USA, 2004.

[63] T. Knott. Smart surrogates. *BP Frontiers magazine*, April 2004.

[64] Y.-B. Ko and N. H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *MobiCom '98*, pages 66–75, New York, NY, USA, 1998. ACM Press.

[65] J.-B. Lee, Y.-B. Ko, and S.-J. Lee. EUDA: detecting and avoiding unidirectional links in ad hoc networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 8(4):63–67, 2004.

[66] S. Lee, B. Bhattacharjee, and S. Banerjee. Efficient geographic routing in multihop wireless networks. In *MobiHoc '05*, pages 230–241, New York, NY, USA, 2005. ACM Press.

[67] W. Lou and Y. Fang. Predictive caching strategy for on-demand routing protocols in wireless ad hoc networks. *Wireless Networks*, 8(6):671–679, 2002.

[68] H. Lundgren, E. Nordstr, and C. Tschudin. Coping with communication gray zones in IEEE 802.11b based ad hoc networks. In *WOWMOM '02*, pages 49–55, New York, NY, USA, 2002. ACM Press.

[69] M. K. Marina and S. R. Das. Performance of route caching strategies in dynamic source routing. In *ICDCSW '01*, page 425. IEEE Computer Society, 2001.

[70] M. K. Marina and S. R. Das. Routing performance in the presence of unidirectional links in multihop wireless networks. In *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 12–23, New York, NY, USA, 2002. ACM Press.

[71] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *MobiCom '00*, pages 255–265, New York, NY, USA, 2000. ACM Press.

[72] N. Meghanathan. Comparison of stable path selection strategies for mobile ad hoc networks. In *ICN/ICONS/MCL*, 2006.

[73] S. Murthy and J. J. Garcia-Luna-Aceves. A routing protocol for packet radio networks. In *MobiCom '95: Proceedings of the 1st annual international conference on Mobile computing and networking*, pages 86–95, New York, NY, USA, 1995. ACM Press.

[74] T. Narten. Assigning experimental and testing numbers considered useful. RFC 3692, `http://www.ietf.org/rfc/rfc3692.txt`, January 2004.

[75] M. Pandey, R. Pack, L. Wang, Q. Duan, and D. Zappala. To repair or not to repair: Helping routing protocols to distinguish mobility from congestion. In *IEEE Infocom MiniSymposia' 2007*, Anchorage, AK, USA, May 2007.

[76] P. Papadimitratos and Z. Haas. Secure routing for mobile ad hoc networks. In *CNDS '02*, 2002.

[77] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *INFOCOM'97*, volume 3, pages 1405–1413, Kobe, Japan, April 1997.

[78] M. R. Pearlman, Z. J. Haas, P. Sholander, and S. S. Tabrizi. On the impact of alternate path routing for load balancing in mobile ad hoc networks. In *MobiHoc '00*, pages 3–10, Piscataway, NJ, USA, 2000. IEEE Press.

[79] G. Pei, M. Gerla, and T.-W. Chen. Fisheye state routing in mobile ad hoc networks. In *ICDCS Workshop on Wireless Networks and Mobile Computing*, 2000.

[80] G. Pei, M. Gerla, and X. Hong. Lanmar: landmark routing for large scale wireless ad hoc networks with group mobility. In *MobiHoc '00*, pages 11–18, Piscataway, NJ, USA, 2000. IEEE Press.

[81] G. Pei, M. Gerla, X. Hong, and C.-C. Chiang. A wireless hierarchical routing protocol with group mobility. In *IEEE Wireless Communications and Networking Conference, WCNC1999*, number 1, pages 1538–1542, New Orleans, LA, USA, September 1999. IEEE, IEEE.

[82] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing. RFC 3561, `http://www.ietf.org/rfc/rfc3561.txt`, 2003.

[83] C. E. Perkins, E. M. Belding-Royer, and I. D. Chakeres. Ad hoc on demand distance vector (AODV) routing. IETF Internet draft, Oct. 2003.

[84] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *SIGCOMM '94*, pages 234–244, New York, NY, USA, 1994. ACM Press.

[85] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *WMCSA '99*, Feb 1999.

[86] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *WMCSA '99: Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, page 90. IEEE Computer Society, Feb 1999.

[87] J. Postel. Internet protocol. RFC 791, `http://www.ietf.org/rfc/rfc0791.txt`, September 1981.

[88] J. Postel. Transmission control protocol. RFC 0793, `http://www.ietf.org/rfc/rfc0793.txt`, 1981.

[89] R. Prakash. Unidirectional links prove costly in wireless ad hoc networks. In *DIALM '99*, pages 15–22, New York, NY, USA, 1999. ACM Press.

[90] K. N. Ramachandran, M. M. Buddhikot, G. Chandranmenon, S. Miller, E. M. Belding-Royer, and K. C. Almeroth. On the design and implementation of infrastructure mesh networks. In *WiMesh '05*, 2005.

[91] T. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall PTR, 2001.

[92] E. Royer and C. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, pages 46–55, Apr. 1999.

[93] N. Sadagopan, F. Bai, B. Krishnamachari, and A. Helmy. Paths: analysis of path duration statistics and their impact on reactive manet routing protocols. In *MobiHoc '03*, 2003.

[94] N. Sadagopan, F. Bai, B. Krishnamachari, and A. Helmy. Paths: analysis of path duration statistics and their impact on reactive manet routing protocols. In *MobiHoc '03*, pages 245–256, New York, NY, USA, 2003. ACM Press.

[95] P. Sinha, R. Sivakumar, and V. Bharghavan. Cedar: Core extraction distributed ad hoc routing. In *INFOCOM '99*, pages 202–209, New York, NY, USA, March 1999. IEEE.

[96] W. Su, S.-J. Lee, and M. Gerla. Mobility prediction and routing in ad hoc wireless networks. *Int. J. Netw. Manag.*, 11(1):3–30, 2001.

[97] J. Sucec and I. Marsic. A query scope agent for flood search routing protocols. *Wireless Networks*, 9(6):623–636, 2003.

[98] F. Tobagi and L. Kleinrock. Packet switching in radio channels: Part ii - the hidden terminal problem in carrier sense multiple-access modes and the busy-tone solution. *IEEE Trans. Commun. COM-23(12)*, pages 1417–1433, 1975.

[99] C.-K. Toh. A novel distributed routing protocol to support ad hoc mobile computing. In *IEEE 15th IPCCC*, pages 480–486, Phoenix, AZ, USA, March 1996. IEEE.

[100] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. *Wirel. Netw.*, 8(2/3):153–167, 2002.

[101] P. F. Tsuchiya. The landmark hierarchy: a new hierarchy for routing in very large networks. In *SIGCOMM '88*, pages 35–42, New York, NY, USA, 1988. ACM Press.

[102] Y. Wang, V. C. Giruka, and M. Singhal. A truthful geographic forwarding algorithm for ad-hoc networks with selfish nodes. *International Journal of Network Security (IJNS)*, 5(3):252–263, 2007.

[103] B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *MobiHoc '02*, pages 194–205, New York, NY, USA, 2002. ACM Press.

[104] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys '03*, pages 14–27, New York, NY, USA, 2003. ACM Press.

[105] M. D. Yarvis, W. S. Conner, L. Krishnamurthy, J. Chhabra, B. Elliott, and A. Mainwaring. Real-world experiences with an interactive ad hoc sensor network. In *IWAHN '02: Proceedings of the International Workshop on Ad Hoc Networking*, Vancouver, Canada, 2002.

[106] F. Ye, A. Chen, S. Liu, and L. Zhang. A scalable solution to minimum cost forwarding in large sensor networks. In *ICCCN '01*, 2001.

[107] X. Yu and Z. M. Kedem. A distributed adaptive cache update algorithm for the dynamic source routing protocol. In *INFOCOM'05*, Miami, FL, USA, March 2005.

[108] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *SenSys '03*, pages 1–13, New York, NY, USA, 2003. ACM Press.

Vita of Rendong Bai

(i) Background

- Date of Birth: Dec 25, 1971
- Place of Birth: Zhuanghe, China

(ii) Academic Degrees

- Master of Engineering in Computer Aided Design, Beijing University of Aeronautics and Astronautics (BUAA), Beijing, China, 1998.
- Bachelor of Engineering in Aircraft Manufacturing, Beijing University of Aeronautics and Astronautics (BUAA), Beijing, China, 1995.

(iii) Professional Experience

- Visiting Assistant Professor, Department of Computer Science, Eastern Kentucky University, August 2007–present.
- Independent Software Vendor (ISV) Program Manager, Red Flag Software Co., Ltd., Beijing, China, 2000.
- Software Engineer, Institute of Software, Chinese Academy of Sciences (ISCAS), Beijing, China, 1998–2000.

(iv) Honors

- University of Kentucky Student Travel Support Award, 2007.
- University of Kentucky Student Travel Support Award, 2005.
- Outstanding Contribution Award, Organizing Committee of the 22nd UPU (Universal Postal Union) Congress, Beijing, China, 1999.
- Excellent Student Awards, Beijing University of Aeronautics and Astronautics (BUAA), Beijing, China, 1991–1992, 1992–1993, 1993–1994, 1994–1995.

(v) Publications

- Rendong Bai and Mukesh Singhal, "DOA: DSR Over AODV Routing for Mobile Ad Hoc Networks," *IEEE Transactions on Mobile Computing (TMC)*, vol. 5, no. 10, pp. 1403-1416, 2006.
- Rendong Bai, Mukesh Singhal, and Yongwei Wang, "On Supporting High-Throughput Routing Metrics in On-Demand Routing Protocols for Multi-Hop Wireless Networks," *Journal of Parallel and Distributed Computing*, vol. 67, no. 10, pp. 1108-1118, 2007.
- Rendong Bai, Mukesh Singhal, and Yi Luo, "Enhancing Performance by Salvaging Route Reply Messages in On-Demand Routing Protocols for MANETs," *Ad Hoc & Sensor Wireless Networks*, 2007, accepted.
- Rendong Bai and Mukesh Singhal, "Route Discovery in Mobile Ad Hoc Networks: From Unilaterality to Bilaterality," *Mobiquitous 2007: the 4th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, Philadelphia, PA.
- Rendong Bai and Mukesh Singhal, "Carpooling in Mobile Ad Hoc Networks: the Case of Multiple-Target Route Discovery," *WiOpt 2007: 5th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, Limassol, Cyprus.
- Rendong Bai and Mukesh Singhal, "BRD: Bilateral Route Discovery in Mobile Ad Hoc Networks," *IFIP Networking 2007*, Atlanta, GA.
- Rendong Bai and Mukesh Singhal, "Salvaging Route Reply for On-Demand Routing Protocols in Mobile Ad-Hoc Networks," *MSWiM 2005: Proceedings of The 8th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, Montreal, Canada.
- Mukesh Singhal, Rendong Bai, Yun Lin, Yongwei Wang, Mengkun Yang, and Qingyu Zhang, "Key Management Protocols for Wireless Networks," technical report, TR 475-07, Department of Computer Science, University of Kentucky, 2007.