Local Repair and Next-hop Backup Route Based improvements in AODV Routing Protocol for Mobile Ad Hoc Networks

by

Mohammed Rizwan Adil

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
August 3, 2013

Keywords: Improvements in on demand reactive routing protocols,
AODV Routing Protocol, Local Repair, Next-hop Backup

Approved by

Thaddeus Roppel, Chair, Associate Professor, Electrical and Computer Engineering
Prathima Agrawal, Samuel Ginn Distinguished Professor, Electrical and Computer Engineering
Chwan-Hwa “John” Wu, Professor, Electrical and Computer Engineering

Abstract

As Mobile Ad hoc Networks (MANETs) continue to experience increasing popularity, several different protocols have been proposed to efficiently transmit data among the participating nodes. These protocols have to be robust and flexible to respond to the dynamic topology and decentralized nature of MANETS. The Ad hoc On Demand Distance Vector (AODV) Routing Protocol is one of the most commonly used reactive protocols for routing information in MANETS. Even though AODV performs well, it suffers from several shortcomings. This thesis aims to modify and upgrade the performance of AODV. Two different schemes are proposed which improve upon different aspects of the standard AODV routing protocol. In the first scheme, the AODV protocol is improved by adding the Local Repair feature. In this protocol, intermediate nodes in existing paths try to find new paths to the destination in the event of a link breakage. In the second scheme, Next-hop Backup Route is introduced. According to this scheme, once a path is established, every upstream node on an active route creates a backup path for its next-hop node. Thus, when link breakage happens, the upstream node can depend upon its backup node to forward the packet to the node that was previously its next-hop. Both these schemes increase the number of data packets successfully transmitted to the destination.

Acknowledgments

Every treasurable moment of my graduate school career has been shared with many people. It has been a great privilege to spend the last couple of years at Auburn University, and the people here will always remain dear to me.

My first debt of gratitude must go to my advisor Dr. Thaddeus Roppel. He patiently provided the vision, encouragement and advice I needed to complete my thesis. Being a strong and supportive advisor, he has always given me freedom to pursue independent work.

Special thanks to my committee members Dr. Prathima Agrawal and Dr. Chwan-Hwa "John" Wu for their support, guidance and valuable suggestions. Their guidance has served me well and I owe them my heartfelt appreciation.

I am deeply grateful to Ms. Marcia Boosinger and Ms. Claudine Jenda for providing me with Graduate Assistantship and financial aid.

My friends in US, India and other parts of the world were sources of joy and support. Special thanks to Swathi Dumpala, Mustafa Shihab and Rathan Raj.

I wish to thank my family back in India for their love and support. I am forever indebted and grateful to them for everything they did for me. Most importantly, I would like to thank God for blessing me with such great friends and family and for making things fall in place.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, mobile computing has enjoyed an unprecedented rise in popularity which has largely been facilitated by the improvements in software and processing power of the devices. At the current pace of technological innovation, the majority of the world's population is expected to be online by 2025. [1] Whether it be laptops, tablets or phones, most wireless devices currently in use directly interact with humans and connect to the global Internet. In the future however, a significant number of wireless devices are expected to work with little or no human guidance and are expected to perform without necessarily connecting to the Internet [2]. These devices are expected to generate short lived networks just for immediate communication needs without any external intervention- in other words, an ad hoc network. An ad hoc network may be understood as the cooperative engagement of collection of mobile nodes without any centralized access or control point in which each node acts as a specialized router [2]. The IEEE 802.11 subcommittee defines an ad hoc network as a wireless network composed of stations within mutual communication range of each other, created in a spontaneous manner and is limited only by temporal and spatial constraints [3]. They are supposed to interact using a dynamic network in which nodes join and leave arbitrarily. There are no restrictions on where the stations can be located with respect to each other and the network is expected to be self-starting.

Ad hoc networks may be used for several personal applications like conferencing, home networking, embedded computing applications etc. Significant strides have already been made in the research, experimentation and deployment of autonomous or semi autonomous unmanned ground [4], air [5,6] and sea [7] vehicles. These vehicles and vehicle system are being used for both domestic and military applications and they use ad hoc networks to communicate among each other [6,7]. On an

industrial scale, these networks can also be used in several applications like search and rescue, agriculture, sensor based analysis and large scale transportation [2].

Ad hoc networks by their very nature help to address some issues that we currently face because of the way the Internet is structured. The evolution of Internet has been such that two devices that are in immediate wireless range of each other still have to use routers and switches at remote locations to forward packets between each other. Ad hoc networks may be able to change this by directly connecting multiple wireless devices without the intermediate Internet. This will reduce the bandwidth consumption of the Internet and also alleviate some security concerns. If both the sender and the receiver are in immediate vicinity of each other, then it becomes difficult to compromise the communication unless the attacker is the same physical space. Ad hoc networks may be employed in campuses, companies and hospitals and may become the first choice for connecting devices that are near by.

Most of these ad hoc networks cannot be dependent on the standard Internet routing protocols because they are not likely to be connected to the Internet all the time. Also, unlike regular Internet networks, ad hoc networks have several additional constraints involved like battery life, mobility and relatively lower processing power. The standard Internet routing protocols exhibit their least desirable behaviors when presented with a highly dynamic interconnection topology [2]. Therefore it becomes impractical to implement the same handful of standard routing protocols in such cases even in the presence of a viable Internet connection. To solve this problem, several different protocols have been designed for ad hoc networks. All these protocols have their own advantages and disadvantages and any particular choice of protocol is a judgment call based on the situation.

It is important to note that even though our intention is to generate an ad hoc network for mobile nodes, these nodes will have to connect to the regular Internet too. For this reason, the standard network architecture is implemented on all these nodes. This includes the four layers of the TCP/IP

protocol and their subsequent functionalities. The only change however is at the layer 3 routing protocol. All the participating nodes have layer 3 IP addresses and assist each other in routing data packets. The layer 2 and layer 1 functionalities work exactly like on the regular Internet and help in transmitting the frames and data.

A very popular ad hoc wireless protocol is the Ad Hoc On Demand Distance Vector (AODV) Protocol. It provides quick and efficient route establishment between two nodes desiring communication. Several researchers have modified and upgraded the AODV depending on their needs. In our particular iteration, we modify the AODV protocol to make it more robust and adaptable for large number of nodes.

1.1 Problem Statement

The goal of this research work is to implement two different schemes to improve the AODV routing protocol. In the first scheme, we study the impact of incorporating Local Repair mechanisms in the AODV routing protocol. In the second scheme, we add backup routes for every next-hop node in a path generated by the regular AODV protocol.

1.2 Thesis Contributions

The contribution of this work is to provide Local Repair and Next-hop Backup Route based improvements for the AODV routing protocol. The core algorithm of the AODV protocol is discussed in detail which is followed by the changes suggested by our improvements. Predictions are made for each of these improvements and compared to the simulation results run in network simulator ns-2.35. It is shown that both Local Repair and Next-hop Backup Route based improvements result in improving

the quality of the AODV protocol by increasing the number of successfully delivered packets and the packet delivery ratio.

1.3 Organization of Thesis

Chapter 2 introduces the reader to different types of traditional routing protocols and compares and contrasts them with each other. This is followed by a discussion on why regular routing protocols are not suitable for ad hoc networks. Following this, a discussion of the Destination Sequenced Distance Vector (DSDV) and the Dynamic Source Routing (DSR) protocols is presented. This is followed by a detailed discussion of the AODV Routing Protocol.

Chapter 3 begins with a discussion of several improvements that have been done on the AODV protocol and justifies our choice of the two selected modifications.

Chapter 4 describes the simulation environment in detail and explains how to implement and analyze the changes in the protocols.

Chapter 5 presents the results of these improvements and compares them to predicted outcomes based on the changes in the algorithm.

Chapter 6 concludes the thesis.

Chapter 2

Literature Review

The TCP/IP Internet Protocol suite has become the de facto standard for all networking communications. Almost every networking device in use today supports TCP/IP. This protocol divides the network into four different layers and defines the purpose of each layer. This division helps simplify the network as different protocols can address different issues at each layer [8].

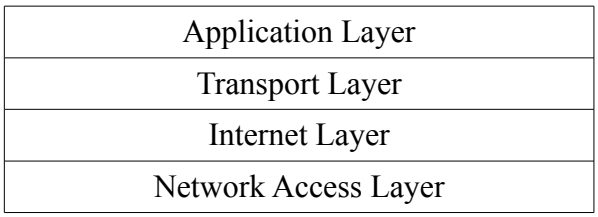

Figure 2.1 TCP/IP Architecture Model

The Application Layer provides an interface between the software running on the computer and the network. It defines the services that applications might need. These services may vary depending upon the user's needs. For instance, some protocols may define the ability to transfer files while others may define the ability to render web pages [10]. Hyper Text Transfer Protocol (HTTP), Hypertext Markup Language (HTML), Post Office Protocol (POP), Simple Mail Transfer Protocol (SMTP) are examples of protocols that operate at this layer.

The Transport layer provides a logical connection between two hosts with functions such as reliable delivery of data, flow control, multiplexing and error-recovery [10]. The transport layer protocols segment the incoming byte stream coming from the application layer and passes it down to

the Internet layer. Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are the two main protocols that are run at this layer. The data unit at this layer is called a 'segment'.

The Internet Layer undertakes the routing process, i.e., it delivers packets from one device to another. The Internet Protocol (IP), which is the foundation of the world wide web operates at this layer. IP delivers the data packets from sender to receiver by using logical addressing and routing protocols [9]. IP assigns logical addresses to all participating nodes which are used to identify a packet's source and destination. All IP packets have a header which contains information about the packet like source and destination addresses, packet length, upper layer protocol etc. The header and hence the packet depends upon the version of IP in use. Routers are used to analyze these data packets and forward them to appropriate destinations [11]. The two versions of IP currently in use are the Internet Protocol version 4 (IP v4) and the Internet Protocol version 6 (IP v6). The routing protocols dynamically learn about groups of addresses in the network and generate routing tables. Routing protocols also enable routers to broadcast its current information to all its neighbors. This process helps to calculate the best possible route to a given destination and to generate routing tables. Routing tables list all the networks that can be reached and the best route to that network. To calculate the 'best route', different protocols use different algorithms and different metrics [10].

The Network Access Layer defines the protocols and hardware required to deliver the data across the physical network. This layer provides services to the Internet Layer. Ethernet is the most common protocol at this layer. This layer defines the protocols for cabling, connectors, voltage levels, wireless frequency bands and physical addressing (MAC level) that are used to deliver data. It can be thought of as comprising of layer 2 link layer and layer 1 physical layer. The data unit at this layer is called a frame.

2.1 Dynamic Routing Protocols in regular networks

Routers add routes to their routing table in three different ways. Directly connected routes are simply added based on their connection, static routes are predefined and dynamic routes are filled up by a routing protocol. This route table is then read by a routed protocol like IP to transmit the packets between different nodes. Routing protocols are a set of messages, rules and algorithms used by routers to learn the routes between different nodes and choose the best route for a particular node [12].

The three main types of dynamic routing algorithms are distance vector, link state and balanced hybrid. The Distance Vector algorithms are based on the Distributed Bellman-Ford algorithm [16]. According to this algorithm, the router learns about all the possible paths to a particular destination. Every path has an associated hop count with it. The router picks the path with the smallest hop count for transmitting data packets and the declares the next-hop on this path as the 'vector'. In essence, the router is oblivious to the data transmission beyond its next-hop. It only knows the number of hops it takes to reach the destination using a particular path.

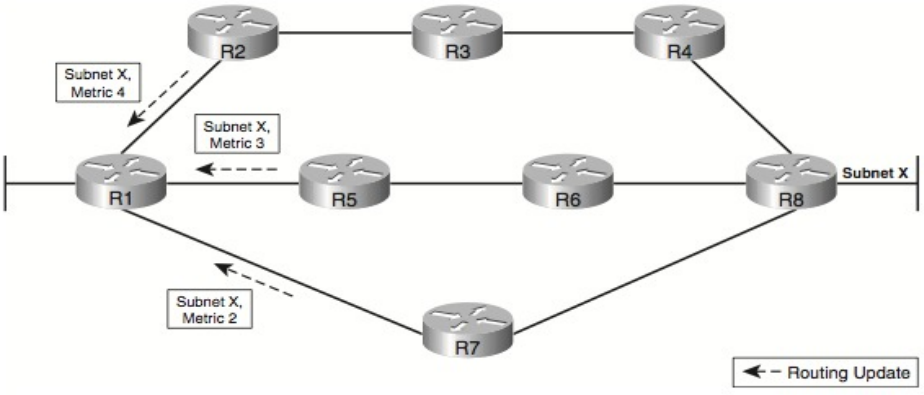

Figure 2.2 Information learned using Distance Vector Protocols [12]

In the above figure, Router R1 choses the Router R7 as its next-hop because it has the least hop count metric. Every router broadcasts information about every destination it can access and the hop count to that destination. Other routers receive this information and process it and edit their routing tables accordingly. Such broadcasts occur on a periodic basis thereby keeping the route information fresh and suitable for use. When a route becomes invalid, the routers declare the hop count metric as infinity and broadcast this information. Other routers then realize that this route has become defunct and remove it from their routing table.

When a distance vector protocol running on a particular router receives an update from a neighboring router, it performs these six steps as enunciated by the Bellman-Ford algorithm [13].

1. Increments the metrics of the incoming routes in the advertisement. In the above example, when R6 receives a routing update from R8, it increments the hop count metric from 1 to 2 before broadcasting it any further.
2. Compares the information in the routing update to what it already has in its routing table. Referring to the above example, lets say that R1 just has information about a 4 hop destination to R8 through R2. If it receives new information for the same destination, it compares it to the already existing 4 hop destination.
3. If the new information is better than the one in the routing table, the information in the routing table is discarded and replaced with the better information. If a router supplies R1 with information to reach R8 that is better than the existing information through R2, the new information takes place of the old information. In our case, both R5 and R7 provide better routes than R2, hence the information through R2 is discarded.
4. However, if a neighbor supplies information that is worse than already existing information, no action is taken. In the example, consider that R7 is discovered first and the routing table is updated with hop count 2. If later on, R2 and R5 are discovered, both of which have higher hop

count than R7. These new routes are simply discarded.

5. If the neighbors information is exactly the same, reset the timer for that entry in the routing table to indicate that the route is still fresh and capable to send packets.
6. If a router receives information about a destination that has the same metric but a different next-hop, it would still be added to the routing table provided it does not exceed the maximum number of equal cost paths. If an R9 is discovered which establishes a two hop path to R8, then both the R7 and R9 paths are considered valid.

The Bellman-Ford algorithm alone does not satisfy all the routing needs. Distance Vector algorithms employ several corrective measures to improve convergence and to avoid routing loops and counting to infinity problems [14, 15]. To advertise changes in the route, triggered updates are used which require immediate broadcasts regardless of the periodic timer. Distance Vector algorithms have a propensity to form loops in the event of link breakage. In some cases, such loops can be avoided by limiting the Time to Live (TTL) on individual data packets. The TTL value is the maximum number of hops a data packet can travel before it becomes void. For example, the Routing Information Protocol (RIP) protocol sets a hop count limit of 15 [13]. To solve the counting to infinity problem and the routing loops it generates, split horizon technique and hold-down timer are used. According to the split horizon technique, if a neighbor router sends a route to a router, the receiving router will not propagate this route back to the advertising router [14]. In the hold-down timer based correction, when a route entry goes down, it is frozen in the routing table for a predetermined 'hold-down time'. This way, the router will not pick up any stray values that may be transmitted by other routers for that particular destination [14].

Link state protocols are based on Dijkstra's Shortest Path First (SPF) algorithm. In contrast to

distance vector algorithms, link state algorithms learn the complete topology of the network. Every router running the Link State algorithm has a picture of the complete network which it then uses to transmit data packets [17]. Link State algorithms generate and multicast Link State Advertisements (LSA). These LSAs are generated only when changes are made in the network and are processed only by devices which are running link state protocols. Unlike Distance Vector algorithms, destination routers reply to the source routers using acknowledgements. As routers learn about routes from LSAs from other routers in the network, they create a topology of the complete network. This topology is represented in the database as a tree which is used to calculate the minimum delay to all the destinations in the network [18]. Link state protocols require greater memory requirements and processing power because they have to maintain a network table, link state database and a routing table.

Hybrid protocols are essentially based on the Distance Vector algorithm but heavily use concepts from the Link State protocol algorithm [13]. Some of the most common hybrid protocols are Routing Information Protocol version 2 (RIPv2), Border Gateway Protocol (BGP) and Enhanced Interior Gateway Routing Protocol (EIGRP).

2.2 Routing Protocols for Ad hoc networks

Routing protocols that operate on regular networks have not been designed to create the self starting and decentralized networks thats are needed for ad hoc networks [2]. Most of these protocols exhibit their least desirable behavior when presented with highly dynamic interconnection topology as in ad hoc networks. It is important to note that the mobile nodes which form ad hoc networks may have much lesser processing power than most laptops or hand-held devices. Therefore, these new protocols should be able to reduce the computational burden without compromising on functionality. These new

protocols should also be able to display better convergence characteristics than regular protocols. Yet another consideration is that ad hoc networks for the most part work through the wireless medium. The problem of routing packets in an ad hoc network may be considered essentially as the distributed version of the Shortest Path problem [20].

We assume our ad hoc network has the following characteristics [2]:

1. All nodes are using IP as the routed protocol and they have IP addresses. These IP addresses are usually static private IP addresses. We cannot run the Address Resolution Protocol (ARP) or the Dynamic Host Configuration Protocol (DHCP) because we are not likely to be connected to the Internet.
2. Not all the nodes are within each other's range.
3. The nodes are mobile such that any particular node may or may not be in the range of some other node.
4. Nodes behave as routers and assist each other in the processing of delivering data packets.
5. Even though some protocols can operate on unidirectional links, we assume all the links in our ad hoc network are bidirectional.
6. Even though protocols can generate and work with multicast and multi-path routes, we focus only on unicast implementations of these protocols.

When it comes to forwarding packets in ad hoc networks, two types of protocols are employed. Proactive or table driven protocols are those in which every node keeps track of all the destinations in the network. Thus, when a node needs to communicate with any destination, it can simply pick up that route from its routing table. The downside of implementing such protocols is that it needs too many control packets- both for periodic updates and also for responding to link breakages. Reactive protocols on the other hand acquire routing information only when it is needed. As a result, they only have to

maintain the routing tables for a short time. This helps in reducing the number of control packets but causes an increase in initial delay time as the protocol is trying to figure out the path.

Ad hoc networks differ from regular networks in several different ways and these differences need to kept in mind when designing protocols for these ad hoc networks. Ad hoc networks do not generally support aggregation techniques like sub-netting or routing prefixes [2]. This makes it difficult to scale the network when new nodes are added. Also, the time for which a node stays at a particular location and the speed of the node negatively impacts the protocol performance. The routing table entries in an ad hoc network change much faster than they do in regular networks [21]. Unlike regular networks, the nodes on an ad hoc network are almost always operating on limited battery power. Therefore, a protocol which processes more number of packets will consume more power. There are several different physical layer wireless standards and speeds which makes it difficult to depend upon any one of them for transmitting packets over multiple hops. The forwarding process should therefore operate only on layer 3. Additionally, these new protocols expose the network to many more security threats.

Several protocols have been proposed for routing packets in ad hoc networks. All these protocols maintain for each destination a preferred neighbor also known as the 'next-hop'. The destination node is identified in every data packet. When a node receives this packet, it forwards it to the next-hop for that particular destination. This process continues until the packet reaches its destination. The generation and maintenance of routing tables differs from one protocol to another. Every protocols focuses on few constraints and based on these constraints attempts to find the optimal path from source to destination. These constraints may be the number of next-hops, bandwidth etc. Similar to regular networks, routing protocols for ad hoc networks can also be classified as either Link State or Distance Vector based. The differences between the link state and distance vector protocols are accentuated in ad hoc networks because of mobility, lower processing power and less storage space.

The Destination Sequenced Distance Vector (DSDV) Routing Protocol was proposed as an ad hoc network adaptation of the Distributed Bellman-Ford algorithm. This is a proactive protocol which performs decently when the number of nodes is neither very low nor very high. To resolve the issue of excessive control packets in proactive protocols, on demand protocols were introduced. The Dynamic Source Routing (DSR) protocol is one of the purest on demand protocols. Thereafter, the Ad hoc On Demand Distance Vector (AODV) Routing protocol was introduced which borrowed some ideas from DSR and created an on demand version of the DSDV.

2.3 Destination Sequenced Distance Vector Routing Protocol

The DSDV routing protocol is basically an adaptive implementation of the modifications to the basic Bellman-Ford algorithm as specified by the Routing Information Protocol making it suitable for dynamic and self starting networks mechanisms as are required for ad hoc networks [19]. It addresses the looping problem presented by the Bellman-Ford algorithm in the face of link breakages and the resulting time dependent nature of interconnection topology describing the links between the mobile nodes. Looping happens in the Bellman-Ford algorithm because nodes are making decisions based on stale and hence incorrect information. This is normally resolved by using some form of inter-nodal coordination protocol [22]. This idea cannot be extended to ad hoc network environments because the nodes have absolutely no correlation between them and the topology is changing rapidly and arbitrarily. The motivation behind DSDV is to retain the simplicity of RIP yet make it suitable enough to avoid looping problems. This is by done by tagging each route table entry with a sequence number so that nodes can quickly distinguish stale routes from the new ones and avoid loop formation.

The basic idea behind the DSDV protocol is that each node in the MANET periodically advertises its view of the interconnection topology with other mobile nodes within the network [19].

This protocol is able to generate and support the changing and arbitrary paths of interconnection. The DSDV protocol works perfectly in the absence of a base or central station but is also compatible to work in the presence of a base station.

2.3.1 Protocol Overview

All packets are transmitted based on the routing tables stored at each node. These routing tables list all available destinations and the number of hops to each destination. Each route table entry is tagged with a sequence number that is originated by the destination node, which is the only new metric proposed in this protocol [19].

Each node periodically transmits updates to keep the information fresh. We assume that these periodic updates are completely independent of other nodes. Mobile nodes have no time synchronization or phase relationship of update periods between mobile hosts. In addition to the periodic updates, new data is immediately generated and transmitted whenever a route change is detected. Both layer 2 and layer 3 addresses are allowed in this protocol.

Data is also kept about the length of the time between the arrival of the first and the arrival of the best route for each destination. Based on this, a decision may be made to delay advertising routes that are about to change. This prevents the advertisement of possibly unstable routes to reduce the number of rebroadcasts of the same destination number.

The figure below shows the movement of a node in an ad hoc network. The mobile host 1 (MH1) moves from one position to another which necessitates the creation of new routes.

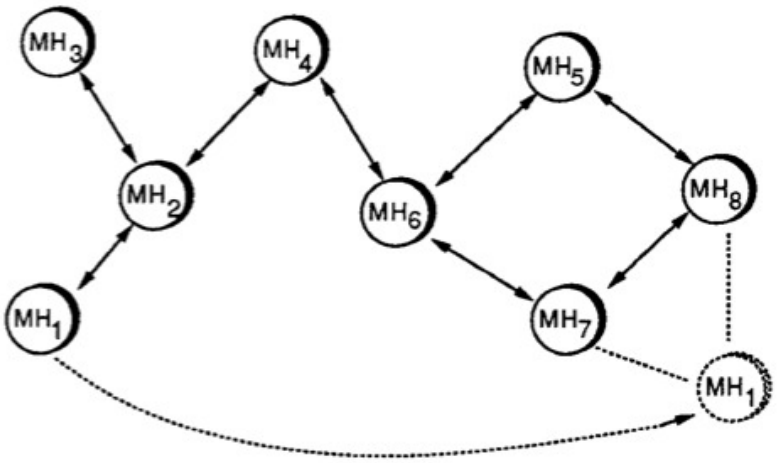

Figure 2.3 Movement of a node in an ad hoc network [19]

2.3.2 Route Advertisement

Each mobile node must advertise its routing table to each of its current neighbors. This advertisement should be made often enough so that all nodes have a record of the recent changes happening in the network. In addition to this, each mobile node agrees to relay data packets to other computers upon request [19].

2.3.3 Route Table Entry Structure

The data broadcast by each mobile computer will contain its new sequence number created by the transmitter, destination address, number of hops required to reach the destination, sequence number of information received regarding that destination as stamped by the destination. The header of this broadcast packet will contain the hardware address and (if appropriate) the network address of the mobile node transmitting it. Routes with more recent sequence numbers are always preferred as the

basis for forwarding decisions. If two paths have the same sequence number, the one with the smallest metric is used. Routes received in broadcast are also advertised by the receiver when it subsequently broadcasts its routing information after adding an increment of one hop to the metric.

It is important to note that wireless links are not always bidirectional. Because of this, the receiving node must check if the transmitting node can also receive the packets, thereby ensuring bidirectional connectivity. Whenever the routing information of a node changes, it immediately issues a rebroadcast. This imposes a requirement on the network to converge as soon as possible. However, if the speed of the nodes is too fast and their relative positions change too quickly, there can be a storm of broadcasts and rebroadcasts which will consume huge amounts of bandwidth [19].

2.3.4 Responding to Topology Changes

Because of the dynamic nature of mobile nodes, several nodes may have broken links or may generate new links at any given instant. The broken link may be detected from layer 2 protocol or may be inferred if no broadcasts have been received for a while. A broken link is described with a metric of infinity i.e. any value greater than the maximum allowed value. When a link to the next-hop is broken, any route through the next-hop is immediately assigned infinity as the next-hop metric and an updated sequence number. This is the only case in which sequence number for a particular destination is generated by a mobile node other than the destination itself. This new sequence number is one greater than the previous sequence number of the destination. This new information is immediately disclosed in a broadcast routing information packet. Later on, if a node which has a direct path to this destination receives this packet with infinity metric, it checks if its direct path sequence number is the same or greater than that of this infinity broadcast packet. If yes, then this triggers a route update broadcast indicating a new path to the destination[19].

To reduce the congestion due to excessive control packets, DSDV defines two types- full dump and incremental dump. As the name specifies, a full dump will carry all of the available routing information where as an incremental dump will only transmit information that has changed since the full dump[19].

The tables 2.1 and 2.2 indicate the forwarding table for Mobile Host 4 (MH4) from the figure 2.3

Destination	NextHop	Metric	Sequence number	Install	Flags	Stable_data
MH_1	MH_2	2	S406_MH_1	T001_MH_4		Ptr1_MH_1
MH_2	MH_2	1	S128_MH_2	T001_MH_4		Ptr1_MH_2
MH_3	MH_2	2	S564_MH_3	T001_MH_4		Ptr1_MH_3
MH_4	MH_4	0	S710_MH_4	T001_MH_4		Ptr1_MH_4
MH_5	MH_6	2	S392_MH_5	T002_MH_4		Ptr1_MH_5
MH_6	MH_6	1	S076_MH_6	T001_MH_4		Ptr1_MH_6
MH_7	MH_6	2	S128_MH_7	T002_MH_4		Ptr1_MH_7
MH_8	MH_6	3	S050_MH_8	T002_MH_4		Ptr1_MH_8

Table 2.1 Forwarding table for MH4 before topology change [19]

Note that after the topology change, the number of hops from MH4 to MH1 changes from 2 to 3. All the sequence numbers also change because significant time has elapsed during the topology changes.

Destination	NextHop	Metric	Sequence number	Install	Flags	Stable_data
MH_1	MH_6	3	S516_MH_1	T810_MH_4	M	Ptr1_MH_1
MH_2	MH_2	1	S238_MH_2	T001_MH_4		Ptr1_MH_2
MH_3	MH_2	2	S674_MH_3	T001_MH_4		Ptr1_MH_3
MH_4	MH_4	0	S820_MH_4	T001_MH_4		Ptr1_MH_4
MH_5	MH_6	2	S502_MH_5	T002_MH_4		Ptr1_MH_5
MH_6	MH_6	1	S186_MH_6	T001_MH_4		Ptr1_MH_6
MH_7	MH_6	2	S238_MH_7	T002_MH_4		Ptr1_MH_7
MH_8	MH_6	3	S160_MH_8	T002_MH_4		Ptr1_MH_8

Table 2.2 Forwarding table for MH4 after topology change [19]

2.3.5 Route Selection Criteria

When a mobile node receives new routing information, that information is compared to the information already available from previous routing information packets. Any route with a better sequence number is used and the route with an older sequence number is discarded. If the new route has the same sequence number but better metric, the existing route is discarded or marked as less preferable. The metrics for routes chosen from the newly received broadcast information are each incremented by one hop. Newly recorded routes are immediately advertised to its neighbors. Those routes with a more recent sequence number are scheduled for advertisement at a later time, which gives time for the newly formed routes to settle [19].

There is no synchronization between different nodes and all nodes transmit packets independently. It may so happen that a node receives newer packets and keep on changing its route even though the destination has not moved. This is because when the node sees a higher sequence number, it changes its route regardless of the metric. After this if it sees a newer route with the same sequence number but with lower metric, it opts for this route. Every new metric is propagated to every mobile node in the neighborhood, which further propagates it to its neighbors and so on. This way, for every change in sequence number, there will be a burst of transmissions. A solution around this problem is to delay the advertisement of routes when a better metric is likely to show up. This way, the route with a later sequence number is available for use but does not have to be advertised immediately, unless it is a route to a previously unreachable destination. This adjustment will create two different routing tables- one for forwarding packets and the other to be advertised.

Even though the DSDV protocol is becoming increasingly obsolete, some of the ideas presented in the DSDV protocol have been incorporated in other advanced protocols like AODV and others.

Perhaps the most significant contribution of DSDV protocol is the use of sequence numbers. Because of excessive rebroadcasts, DSDV performs very poorly as the mobility increases [22]. The throughput of DSDV is very low and the control overhead is very high when compared to reactive protocols [23]. However, DSDV performs well if the nodes move relatively slowly and when the topology changes are not huge [24]. Yet another advantage is that individual packets do not have to wait to start transmitting like in reactive protocols.

2.4 Dynamic Source Routing Protocol

The Dynamic Source Routing protocol is a simple and efficient on demand routing protocol designed specifically for multi-hop ad hoc wireless networks with mobile nodes. Like most other ad hoc routing protocols, it is completely self organizing, self starting and doesn't need any administration or control. The source takes over the responsibility of establishing the route before any packets are sent. Each data packet has in its header the complete list of all the nodes through which it is supposed to pass. This feature avoids the need for periodic updates and makes the routing trivially loop free. This protocol is made up two mechanisms- Route Discovery and Route Maintenance which work together [25]. Unlike DSDV, there is no need for up to date routing information through which the packets are forwarded. Also, the nodes that forward or overhear the packet can cache the routing information for future use. The DSR reacts to changes in the links on only those nodes which currently form a part of its route. It is oblivious to link changes elsewhere in the network.

Certain assumptions are undertaken when implementing the Dynamic Source Routing protocol [25]:

1. All nodes in the ad hoc network are willing to participate fully in network protocols.
2. The diameter of the network, i.e., the minimum number of hops required to transmit data from

one end to another of the network is usually small- between 5 to 10.

3. The speeds with which the nodes move is moderate with respect to transmission latency and wireless transmission range of the underlying network hardware.
4. Nodes may be able to enable promiscuous receive mode according to which hardware delivers the received packet to network driver software regardless of the MAC address.
5. Only one IP address is allowed per node.

Dynamic Source Routing does not use any periodic routing advertisements, link status sensing or neighbor detection packets nor does it depend on any underlying protocols to provide these functions. Both Route Discovery and Route Maintenance are performed on a purely on demand basis.

Route Discovery is used only when a source needs to send a packet to a destination but does not already know a route to it. Route Maintenance is used by the source to detect if the network has changed so that it can no longer use the route to the destination. If the source route is broken, the source can attempt to use any other routes it knows to the destination. If there are no alternate routes, it simply invokes Route Discovery once again. When all nodes needed for current communication have been discovered and all nodes are stationary with respect to each other, the number of overhead packets scales down to zero [25].

In response to a single route discovery, a node may learn and cache multiple routes to any destination. This allows the reaction to routing changes to be much more rapid because when one route fails, the source can try another cached route to the same destination. DSR also allows the use of unidirectional links to be used when necessary to improve the overall performance and network connectivity in the system [25].

2.4.1 Route Discovery

When a node originates a packet for a particular destination, it places in the header of the packet sequence of all the hops that the packet should follow. The source obtains a suitable source route by checking its route cache of routes previously learned. If however, there are no routes in this cache for the desired destination, it initiates a Route Discovery.

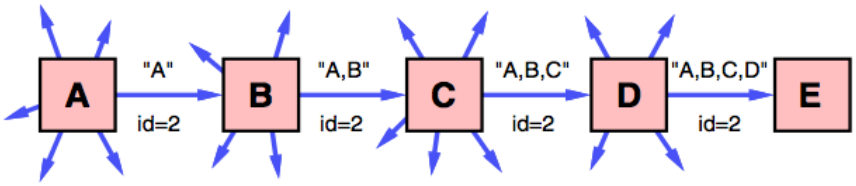

Figure 2.4- Route Discovery example with node A as the initiator and node E as the target [25]

The above figure illustrates an example of the Route Discovery in which node A is trying to find a route to node E. The node A transmits a single local broadcast packet called the Route Request packet which is received by all the nodes within the wireless transmission range of A. Each Route Request message identifies the initiator and target of the Route Discovery and also contains unique Request ID which is determined by the initiator of the request. Each Route Request also contains a record listing the addresses of each intermediate node through which this request has been [26].

When another node receives the Route Request packet, if it is the target to the Route Discovery, it returns the Route Reply to the Route Discovery initiator giving a copy of the accumulated route record from the Route Request. When the initiator receives this Route Reply, it caches this route in its route cache for use in sending subsequent packets to this destination. However, if the node receiving the Route Request recently saw another Route Request from this initiator bearing the same Request ID or if it finds that its own address is already listed in the route record of the Route Request packet, it

discards the request. If not, it appends its own address in the route record of the Route Request message and propagates it by transmitting it as a local broadcast packet with the same request ID [26].

The Route Reply can be sent in several different ways. In the above case, E can initiate Route Discovery for A to find a return path. This mechanism is beneficial if we are operating with unidirectional links. In such as case, E piggybacks this Route Reply to its Route Request packet. Such a process can consume huge control overhead and is therefore best avoided. Therefore, we restrict the operation of DSR only to bidirectional links. Node E simply reverses the route record it finds in the Route Request packet originating from A and use it a source route on the packet carrying Route Reply [26].

Before initiating the Route Discovery, the sending node saves a copy in the local buffer called the Send Buffer. The send buffer packets have a timer associated with them. If the route for this packet is not found within this time, then it is simply dropped. This send buffer employs a First In First Out (FIFO) rule for all its packets. As long as the packet is in the buffer, the node initiates several attempts at Route Discovery. To reduce the overhead of control packets because of multiple Route Discovery attempts, we use exponential backoff to limit the rate at which Route Discovery packets are broadcast [25].

2.4.2 Route Maintenance

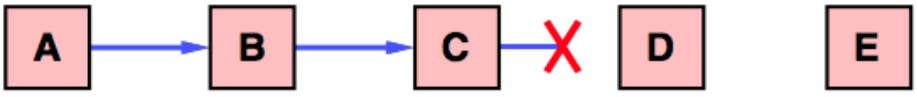

Figure 2.5 Route Maintenance in DSR [25]

Each node transmitting the packet has a responsibility to confirm that the packet has been received by

the next-hop along the source route. Each node can retransmit the packet up to a certain number of times after which it completely gives up. In the above case, A is responsible for delivery to B, B to C and so on. B has to provide acknowledgement to A that it did indeed receive the packet, C to B, D to C and so on. This acknowledgement may be obtained either as an existing standard part of MAC layer or by passive acknowledgement, i.e., B overhears C trying to transmit the packet to D. This acknowledgement is another reason why it is better to have bidirectional links. If we only have unidirectional links, these acknowledgements will have to find a route back to the previous hop which further increases the control packets. If a node makes the maximum possible number of attempts to transmit the packet to the next-hop but still cannot do it, it initiates a Route Error message. In the above case, C initiates Route Error packet back to A indicating that the link C-D is broken. A suspends this broken link and then tries to find if it has any other route to E. It it does, it uses this route to transmit the packets, if not, it initiates another Route Discovery [26].

2.4.3 Additional Route Discovery and Route Maintenance features

When a node forwards or overhears information from other data or control packets, it saves this information for future use. This includes the source route in data packets, route record from Route Request packets and the route in Route Reply packets.

If an intermediate node has a cached route to a destination, it returns a Route Reply to the source. It appends its list of nodes to the destination to the sequence of nodes in the source route in the Route Reply packet. The only condition is that it does not contain there should be no common nodes among the nodes in the source route and those in the intermediate node cache other than the intermediate node itself.

If there are multiple intermediate nodes which have a path to the destination, then they can all

generate replies to the source and create a Route Reply storm. In the case shown in the figure, all the nodes from B-F have a path in its route cache for G.

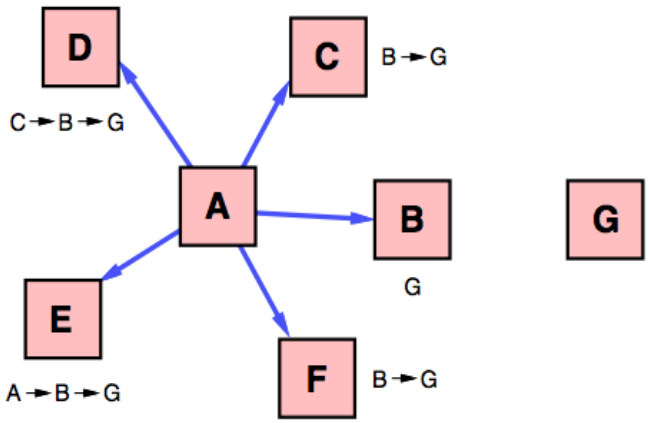

Figure 2.6 RREP storm in DSR protocol [25]

In such a case, individual nodes wait for a random amount of time and overhear any route packets being sent from source to destination. If the route through this intermediate node has lesser number of hops compared to that in the source route of packets being sent to the destination currently, it sends its Route Reply to the source.

The number of hop requests for the Route Request packets are also controlled. Initially, we try to find if the destination is near the node by setting a reasonable hop count like 5-10. If this does not work, a Route Request with no hop limit is sent. This scheme has also been employed with a gradual increase in the hop count for Route Request packets [25]. Such an approach decreases the number of control packets in the network.

Route maintenance also offers several features that help in increasing the efficiency and delivering packets with lesser delay. The node may salvage the packet after returning RERR if it finds another route to the destination in its route cache. If nodes can overhear each other, then they cooperate

and shorten the route. This reduces the number of hops from the source to the destination and makes the route more robust.

Compared to DSDV, DSR requires much lesser overhead as it does not concern itself with maintaining routes to all the possible destinations. Every overhead packet needs processing power, therefore for mobile nodes which operate on limited battery power, DSR is much more preferable to DSDV [27]. DSR stores multiple routes to a destination and exploits the route cache feature very well by saving the route records from data packets and overhearing route records from other data packets [28]. This feature makes DSR very flexible and greatly increases the chance of packet delivery. Unlike other protocols, the source node in DSR controls the packet delivery from start to end thereby enhancing accountability.

2.5 Ad Hoc On-Demand Distance-Vector Routing Protocol

The AODV protocol may be considered as an on demand adaptation of DSDV protocol which also heavily borrows from the DSR protocol. It was specifically designed for ad hoc networks and provides quick and efficient route establishment between nodes desiring communication [29]. It establishes a path between the nodes with minimal control overhead and minimal route acquisition latency.

Compared to the DSDV protocol, the AODV protocol is a significant improvement in several aspects. In DSDV, a single link breakage or a new neighbor can trigger system wide broadcasts. In AODV, if the link status does not effect the ongoing communication, no broadcast occurs. As a result in AODV, local impact only has local effects but in DSDV, local impact has global effects. In AODV, the only non local effects result from a distant source trying to use a broken link. Only those nodes that

have been using the broken link are intimated of its breakage, all other nodes are oblivious to it.

AODV has several minimalist features which help in reducing the number of overhead packets significantly. It does not add any overhead to packets carrying the data and if the routes are not being used, they are simply discarded. This has the benefit of not having stale routes and also helps reduce control packets by avoiding route maintenance. AODV also minimizes the number of routes between any active source and destination to one. While it is possible to have more than one routes, the book keeping and route maintenance for each of these routes is a nontrivial matter. Also if multiple routes to the same source-destination pair have the same broken link, then the purpose of having multiple routes is simply defeated. Although AODV with multiple paths has been proposed [32], for our work we stick only to the single path implementation. The unicast AODV implementation can be extended to generate a multicast implementation just by making a few changes in the protocol [31]. All these features make AODV simple to implement and powerful and robust to respond to ad hoc networks.

2.5.1 AODV Protocol overview

AODV attempts to locate routes for a particular destination only when needed and the route is maintained only as long as necessary. It borrows the idea of sequence numbers from DSDV, which ensures loop freedom. Every node maintains a monotonically increasing sequence number which increases each time it learns of change in topology of its neighborhood. These sequence numbers ensure that the most recent route is selected whenever route discovery is executed. AODV utilizes only bidirectional links between neighboring nodes. This is the only aspect of the physical layer that AODV depends upon. AODV can perform both on wired and wireless media even though it is ideally designed for wireless media. Route tables are used to store the destination and next-hop IP address as well as destination sequence number. Additionally, for each destination, the node maintains a list of precursor nodes which route through it in order to reach the destination. This list is maintained for the purpose of route maintenance in the event of a link breakage. Each route table entry has a lifetime associated with

it which is updated whenever a route is used. If a route has not been used within its lifetime, it is expired. This makes sense because a route not being used is not being maintained which indicates that the nodes along the route most likely have moved. When a link breaks, Route Error messages are sent which provides for quick deletion of invalid routes. There is no additional overhead on data packets because it does not utilize source routing.

2.5.2 Route Discovery

Route Discovery in AODV is a purely on demand process. The discovery cycle comprises of two parts- Route Request and Route Reply. When a node seeks a path, it generates a Route Request message, when another node receives it and sends a reply, Route Reply message is sent back.

Route Discovery process may be described as follows:

1. When a node seeks a route to a particular destination, it broadcasts a Route Request (RREQ) packet.
2. Any node with a current route to the destination can unicast a reply back to the source node. This reply is sent as a Route Reply (RREP) packet.
3. Route information is maintained by each node in its route table.
4. Information obtained through RREQ and RREP is stored alongside other routing information in the same routing table.
5. Stale routes can be eliminated by analyzing their sequence number.
6. Routes with old sequence number are aged out of the system based on their lifetime.

The first thing a node does when it wishes to send a packet is to check if its own route table has a current route to that node. If so, it simply forwards the packet to the next-hop for that particular destination. If however, there is no valid route for that destination, a Route Discovery process is

initiated. The source node creates an RREQ packet. This packet contains the following parameters:

1. *Source node's IP address*
2. Source node's sequence number
3. Destination IP address
4. Last known sequence number for the Destination
5. *Broadcast ID*
6. Time To Live (TTL)

The Broadcast ID is incremented each time each time source node initiates an RREQ. Together the source node IP address and broadcast ID form unique identifier for the RREQ. Once the RREQ is generated, the source node broadcasts it and sets a timer to wait for a reply.

When a node receives an RREQ, it performs these steps in a sequential manner:

1. Check the Source IP and Broadcast ID pair to make sure if it did not already receive this packet earlier. Each node maintains a record of Source IP and Broadcast ID addresses for each RREQ it receives for a limited time. If this is a packet it has already received, it simply discards this packet and does nothing.
2. If however, a particular node receives a RREQ packet for the first time, the node sets up a reverse route entry for the source node in its route table. This entry contains the source node's IP address and sequence number as well as number of hops to the node as well as the IP address of the neighbor from which next-hop was received. This is because the node should know how to forward the RREP if it is received later on. Once it does this, it increments the hop count in the RREQ and rebroadcasts its neighbors.

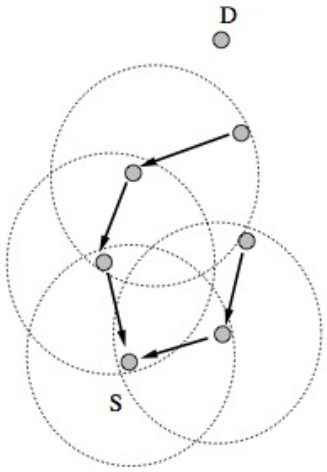

Figure 2.7 Reverse path formation in AODV [29]

In the above figure, we see an instantaneous picture of different Route Replies being generated for the source which is trying to find a route to D. Each of these reverse route entries is maintained for a specified amount of time. If there is no RREP within that time limit, this reverse route entry is simply discarded to prevent stale routing information from lingering in the route table.

3. A node can respond to an RREQ in one of two cases:

   i. The node has an unexpired entry to the destination in its route table and the sequence number associated with this destination must be at least as great as that in the RREQ. This is to ensure that the path to the destination is at least as fresh as the RREQ packet. Such a mechanism prevents formation of loops.

   ii. The node is the destination.

4. Once it is confirmed that a node can reply to the destination, it sends back a unicast RREP packet to the source.

5. If the RREQ is lost, the source node is allowed to retry broadcast of Route Discovery. After a certain number of attempts, we stop sending RREQ and assume that the destination is

unreachable. This number of attempts is usually limited to 2 [31]. In the first attempt, we use a limited hop count by setting a lower TTL value. This is because we want to be able to find the destination without clogging the network with too many rebroadcasts of the RREQ packet. If this RREQ fails, we rebroadcast the packet with an increased TTL value. This process continues until the maximum number of retries is exhausted. When a route is established, the distance to the destination is recorded is recorded in the route table. Next time this same destination has to be accessed, the TTL value is set to what it was before plus a small increment.

When a node determines that it can respond to an RREQ, it generates an RREP. This RREP varies slightly depending on which node is generating it.

i. If the destination node is generating the RREP, it places its current sequence number in the packet, initializes hop count to zero and places the length of time this route is valid in the RREP's lifetime field.
ii. If an intermediate node is generating the RREP, it places its record of the destination sequence number in the packet and sets the hop count equal to the distance from the destination. It also calculates the amount of time for which its route table entry for the destination will still be valid.

In both these methods, the RREP is unicast towards the source node using the node from which it receives the RREQ as the next-hop.

When an intermediate node receives an RREP, it sets up an *forward path* entry to the destination in its route table much like the reverse path entry with RREQ packets. This entry contains the IP address of the destination and the IP address of the neighbor from which the RREP arrived and the hop

count to that destination. The distance from this intermediate node to the destination is hop count plus one. This forward path entry also has a lifetime which is set to lifetime contained in the RREP. Each time the route is used, its associated lifetime is updated. If the route is unused within the specified lifetime, it is simply deleted. After processing the RREP, the node forwards it to source.

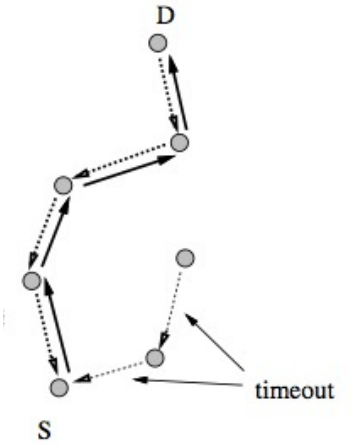

Figure 2.8 Forward path formation in AODV [29]

If a node receives a second RREP for/from the same destination, it checks if this new RREP has a better destination sequence number or smaller hop count than the one it previously processed. If yes, this new RREP is forwarded. In all other cases, these RREPs are simply dropped. This mechanism decreases the number of RREPs navigating towards the source and also keeps the routing information up to date. The source begins data transmission as soon as the first RREP is received and can update later if it finds a better route.

2.5.3 Route Maintenance

The source nodes decides how long to maintain a particular route once it has been finalized. This route from the source to the destination is referred to as the active path. Movement of any routes

outside this active path does not impact the route maintenance.

Several different scenarios arise for route maintenance.

1. If source node moves, we have to reinitiate Route Discovery. This is not such a big problem as it is likely that there will be an existing forward path to the destination from some of the nodes near the source node.
2. When either the destination or an intermediate node moves, a Route Error (RERR) message is sent to the source node. This RERR message is initiated by the node upstream of the break. An upstream node is used to refer to a node that is closer to the source node.
3. The Route Error packet lists each of the destinations that are now unavailable because of this breakage. If there is at least one node between the upstream node and the destination, the upstream node also broadcasts the packet in addition to unicasting it to the source. When the neighbors receive this RERR, they mark their route to the destination as invalid by setting the distance to the destination equal to infinity and in turn propagate the RERR to their own precursor nodes, if any such nodes are listed for the destination in the route table.
4. When the source node receives the RERR, it checks if it still needs Route Discovery. If yes, then it initiates it.
5. Route entries with an infinity metric are not immediately deleted because they contain useful routing information with a recent destination time stamp. They expire in roughly the same amount of time as do reverse routes formed during route discovery. Discarding current route information even of negative variety is not suggested because it can help in taking some decisions later on.
6. If a node receives a data packet destined for a node for which it does not have an active route, it creates a route error message for the destination node which is broadcast and also sent back to the source. In this way, the node has informed its upstream node that it should stop sending data

packets.

2.5.4 Local Connectivity Management using Hello packets

Each time a node receives a broadcast from a given neighbor, it updates the lifetime associated with that neighbor in its routing table. If there is no entry for that neighbor in the routing table, the node creates one. If a node has not broadcast anything within the last hello interval, it can broadcast a Hello packet to its neighbors to inform them that it is still in its vicinity. Hello interval is defined as the maximum amount of time before the node can broadcast hello packet. This is usually set to 1 second. Hello message in an RREP message that contains node's IP address and current sequence number. It has TTL value of 1 thereby preventing it from being rebroadcast. Hello messages are incorporated so that AODV does not rely on any underlying protocols for connectivity information.

AODV has emerged as a robust and powerful protocol to be used in ad hoc networks. Even though there are cases in which DSR performs better than AODV, AODV is considered a better protocol because it scales well to large number of nodes [31]. Also, AODV performs better when nodes move more frequently [32]. The overhead packet consumption for a smaller number (20-100) of nodes is much more for DSDV when compared to AODV [33]. Even though the choice of protocol depends a lot on the scenario which is defined by the number of nodes, speed, mobility etc, in most cases AODV performs better than AODV.

Chapter 3

Enhancements for the AODV Routing Protocol

The AODV routing protocol has been enhanced, optimized and improved in several ways depending upon various parameters. Some researchers have tried to focus on some intrinsic aspects of AODV like Local Connectivity [37], Local Repair [38, 41] and security where as others focus on external aspects like energy efficiency [39] and node density.

AODV is expected to perform at varying speeds of nodes and varying pause times. The pause time here is defined as the time for which a particular node remains stationary after moving to a new destination. The greater the speed, the faster scenarios change and new routes need to be created. The lesser the pause time, the quicker the response has to be for finalizing new paths.

There are several metrics with which we measure the success of an enhancement or an improvement of a routing protocol. Some improvements focus on increasing the number of packets delivered, some others focus on reducing the end to end delay time and some just focus on reducing the overhead. Depending on the scenario, one or more of these features may be more desirable.

3.1 Scheme 1- Local Repair based improvement for the AODV Routing Protocol

In the regular AODV protocol, after a route has been established and been declared as an active route, if a link breakage happens, the node upstream of the break creates a Route Error message listing all the destinations which have become unreachable due to the break. If instead of sending an error message to the source node, if the upstream node attempts to repair the broken node itself, less number of data packets will be lost and the route may be restored with a lower overhead. Also, the source node is not at all bothered with another Route Discovery process.

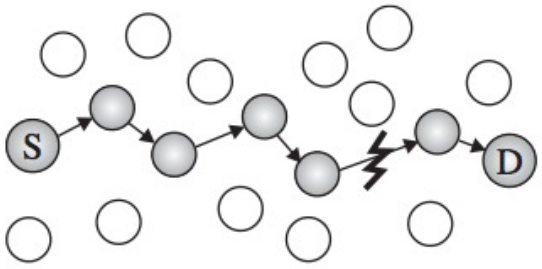

Figure 3.1 Link breaks in an active route [40]

For smaller routes, Local Repair is not expected to show much advantages but for larger routes, especially with 10 or more than 10 hops, Local Repair is extremely beneficial. This is because in larger routes, the links are expected to break more often and if the intermediate nodes always keep sending Route Error packets to the source which in turn keeps initiating Route Discovery, huge number of control packets are consumed and the performance will deteriorate.

Local Repair makes the node upstream of the break to attempt a repair of the route. This is done by broadcasting a Route Request with a TTL set to the last known distance of the destination plus an increment value. This TTL value is used with an assumption that the destination is not likely to be far away from where it was before the break.

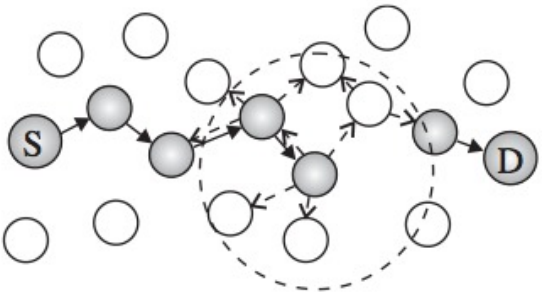

Figure 3.2 Intermediate node broadcasts RREQ packets [40]

This node increments the sequence number of the destination in the RREQ packet by 1 before transmitting it. This prevents the nodes further upstream of this node from replying to the RREQ. Thus, this mechanism prevents loop formation.

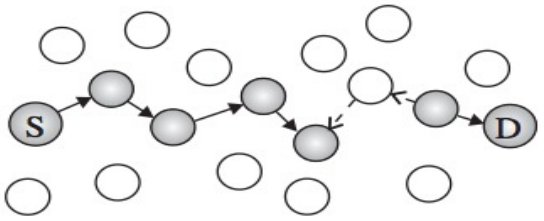

Figure 3.3 New route is being created after receiving RREP [40]

Note that this RREQ broadcast is done only once. If no node replies to this broadcast, the intermediate node simply sends a RERR back to the source node.

There are several important features of the Local Repair improvement in the AODV Routing Protocol which need to be discussed:

1. Local Repair helps increase the number of data packets that reach its destination.
2. As the network size increases, it becomes more and more difficult for AODV to deliver packets to their destinations. The path length of these routes increases and and a single route may have multiple points of breakage. In Local Repair based AODV, we initiate the repair from an intermediate node which is nearer to the destination than the source, hence routes are expected to get repaired quickly and with lesser overhead.
3. Local Repair based AODV may cause longer paths to the destination than regular AODV. This is because the intermediate nodes do not change the source to intermediate node path at all. They try to generate a path from the intermediate node to the destination regardless of the destination's new position with respect to the source. There could be an example in which the destination is very close to the source but because of local Repair, packets may have to traverse a longer route path. The source will not know about the nearby destination until it receives a

RERR and performs a fresh Route Discovery.

4. Local Repair AODV reduces the number of RREQ transmissions and therefore reduces the control overhead of the protocol. The ratio of number of all packets (RREQ, RREP and RERR) per data packet is lower for Local Repair AODV than for regular AODV.
5. The end to end delay for Local Repair AODV is expected to be much lower than that of regular AODV. This is because the in regular AODV, when a break happens, the RERR packet has to travel to the source and the source has to initiate a RREQ for the destination. This process takes much more time than Local Repair AODV in which an intermediate node transmits RREQ to initiate a route reconstruction.

3.2 Scheme 2- Next-hop Backup based improvement for the AODV Routing Protocol

Several backup based improvements have been proposed for AODV. The challenge with backup based routes is to avoid loops and keep the number of control packets in check. Multi-path AODV reduces overhead packets by about 20% and improves the end-to-end delay by more than a factor of 2 [34]. Most of the backup schemes focus on generating a backup route for the source-destination pair. In our case, we propose a backup node for every link in the active route. This is accomplished by proposing two simple control packets which are used to identify the backup node.

When a link breaks, the upstream node transmits the packet to this backup node which in turn transmits it to its ex next-hop. To understand this scheme, please refer to the forward path set up in the regular AODV protocol. When a source needs to communicate with a destination, it sends out an RREQ, this RREQ creates a reverse route as it progresses until we find a path to the destination or the the destination itself. When the destination is located, an RREP is unicast back to the source through all

the intermediate nodes.

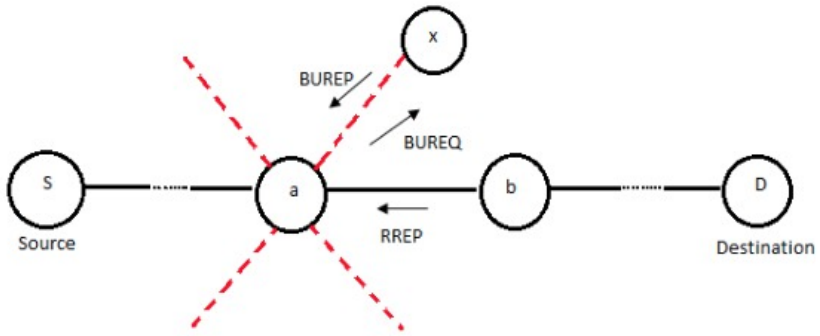

Figure 3.4 Node *a* locating a backup node *x* for a node *b* on its active path

In the above figure, there are *n* intermediate nodes i_1 to i_n. The RREQ is sent along the path from source to destination and the RREP is sent along the path from destination to source. Consider the two intermediate nodes in the figure *a* and *b*. The node *b* is sending an RREP to *a* which it received either from the destination or from some other intermediate node that has route to the destination.

According to our scheme, the moment node a receives this RREP, it tries to locate a backup node for the *a-b* link. It sends out a special packet called the Backup Route Request (BUREQ). This BUREQ is sent out only once, therefore it does not need any sequence numbers. This packet has *a's* IP address, *b's* IP address, destination IP and sequence number and a TTL of 1. Through this BUREQ, we are trying to locate a node *x* such that node *x* can become a backup node for *a-b* link.

The Backup node establishment is undertaken as follows:

1. When a node receives this BUREQ, it checks the destination IP address and sequence number and checks its own route table to see if it has a route to this destination. If it has an active route to the destination, it drops this BUREQ packet.

   The node *b* itself gets this request and it drops the BUREQ packet because it has a direct path to the destination.
2. If some other node receives the BUREQ and it does not have a path to the destination but has a path to *b*, it is eligible to be a backup node.
3. All such nodes which are eligible to be a backup node reply with a Backup Reply (BUREP). These nodes add extra rows to their routing table declaring themselves as Backup nodes for a-b link.
4. When node *a* receives the first BUREP, it registers this node as a Backup node for *b*. The node *a* adds this information to its routing table.

When the primary link a-b gets broken, four cases may arise.

Case 1:

Check *a*'s route table for a backup entry to *b*. If there is no entry, send RERR back to the source.

Case 2:

If there is a backup node listed in *a*'s route table, send the packet to the backup node. This backup node receives the packet and checks its own routing table. The backup node will forward all the packets it receives for the destination to the node *b*.

Case 3:

The node *a* checks its routing table and forwards the packet to *x*. By now, the node *x* has moved away and the packet does not reach *x*. Node *a* sends a RERR back to the source.

Case 4:

The packet reaches x but its route table entry has expired or become corrupted and it cannot send a packet to to b. Node x sends a RERR back to the source.

Chapter 4

Simulation

All the simulations are done using network simulator 2 (ns2). ns2 provides support for TCP and routing over wired and wireless networks [44]. Once ns2 is downloaded and installed, it contains the C++ files for several different wired and wireless protocols from all the layers in its repository. To implement a particular simulation, we write a TCL file to select which protocols we want to use from this ns2 collection. The TCL file is implemented on a scenario file. Scenarios are simulated environments generated by ns2. When the user supplies information like the number of nodes, the size of the room, simulation time etc, scenario files get generated. These scenario files can be saved and different protocols or protocol modifications can be run on them. This way, changes in the code can be measured and studied.

4.1 Scenario file generation in ns2

In our case, we generate several scenario files to measure the performance of our protocols. It has been suggested that the average number of neighbor nodes should be maintained between 6-8 for better performance and scalability [35,40]. The area within which these nodes are allowed to navigate is referred to a the room size. This is described by the length and breadth of the enclosure within which the nodes can move freely. Keeping this in mind, we design our scenarios for different number of nodes as follows:

No of nodes	Room size
10	500 x 500
50	1000 x 1000
100	1500 x 1500
175	2000 x 2000

250	2500 x 2500

Table 4.1 Scenario generation for testing Local Repair AODV

In addition to the these parameters, we have provided a simulation time of 300s, pause time of 3s and the speed of nodes is set at 5m/s.

However, for the Next-hop Backup Route based improvement, we keep the room size constant and the number of nodes limited. The room size is set to 1000 x 1000 and the number of nodes is increased from 10 to 100. This is done to minimize the number of multiple breakages. In the event of multiple breakages, even after the repair, the packet will have nowhere to go.

No of nodes	Room Size
10	1000 x 1000
30	1000 x 1000
50	1000 x 1000
70	1000 x 1000
90	1000 x 1000
100	1000 x 1000

Table 4.2 Scenario generation for testing Next-hop backup based improvement in AODV

4.2 TCL file specifications

The individual aspects of the scenario implementation as described in the TCL file are listed as follows

Channel	Wireless Channel
Propagation model	Two Ray ground
Network Interface type	Wireless
MAC layer	802.11
Buffer type	FIFO
Size of the buffer	50 packets
Antenna type	Omnidirectional

Bandwidth of each node	1MHz
Packet size	1024 bytes
Data generation rate	512 kbps

Table 4.3 TCL file specifications

The TCL file when run for a particular scenario generates two new files of the type 'trace' and 'nam'. These two files contain the information about the implementation of the protocol described in the tcl on the given scenario file. Individual details like the number of sent and received packets, end to end delay etc can all be extracted from these two files.

4.3 AODV implementation on ns2

In the TCL file, when the user configures AODV, the pointer moves to start and this start moves to command function of the AODV protocol. The AODV protocol then gets implemented on the scenario file using the TCL file. Therefore, if we make changes in the AODV protocol, we will have to recompile ns-2 and rerun the TCL files for the same scenario. The TCL and scenario files are considered as the 'front end' code where as the AODV files are considered 'back end' code.

Before making any changes, it is important to note that the default AODV as provided by ns-2 has its hello packets disabled. The user must enable the hello packets before implementing any enhancement or even running the regular AODV. For every modification of AODV, the front end code stays the same whereas the backend code needs to be changed to implement any of our suggested improvements.

Chapter 5

Results

Scheme 1 – Local Repair based improvement for the AODV protocol

We first compare the Local Repair scheme with the regular AODV protocol. After the incorporation of Local Repair, the protocol shows an average improvement of around 15 percent. Because these scenarios are randomly generated and their topology can vary immensely from scenario to scenario, the performance of the new protocol may differ significantly from scenario to scenario.

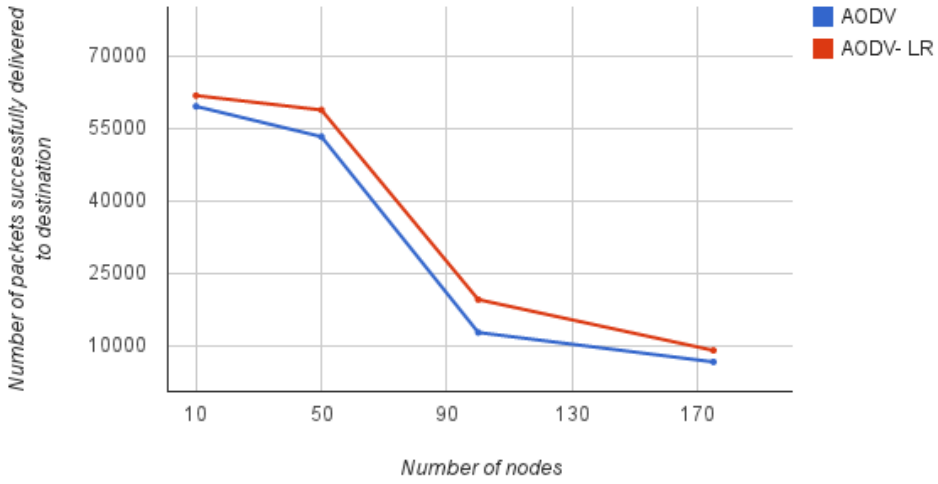

Figure 5.1 Number of packets successfully delivered to destination

The above graph is obtained by calculating the aggregate of all the received packets in the network. AODV with Local Repair performs better because in Local Repair the nodes closer to destination than the source initiate route discovery and hence the path is resolved quickly. This results in less number of packets dropped.

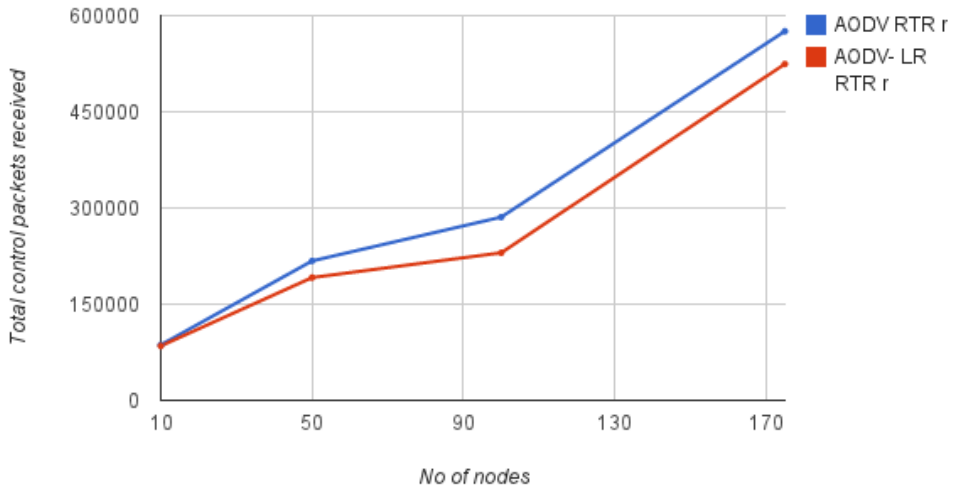

Figure 5.2 Number of control packets

In regular AODV, RERR is initiated as soon as a link breakage is detected. This RERR travels back to the source which initiates a new route discovery.

In AODV with Local Repair, RERR packets are eliminated in some cases as the intermediate node itself initiates a route discovery. Also, the Route Discovery from an intermediate node to the destination takes less number of control packets when compared to that from the source.

In accordance with our expectations, the number of control packets for AODV with Local Repair are always less than that of regular AODV.

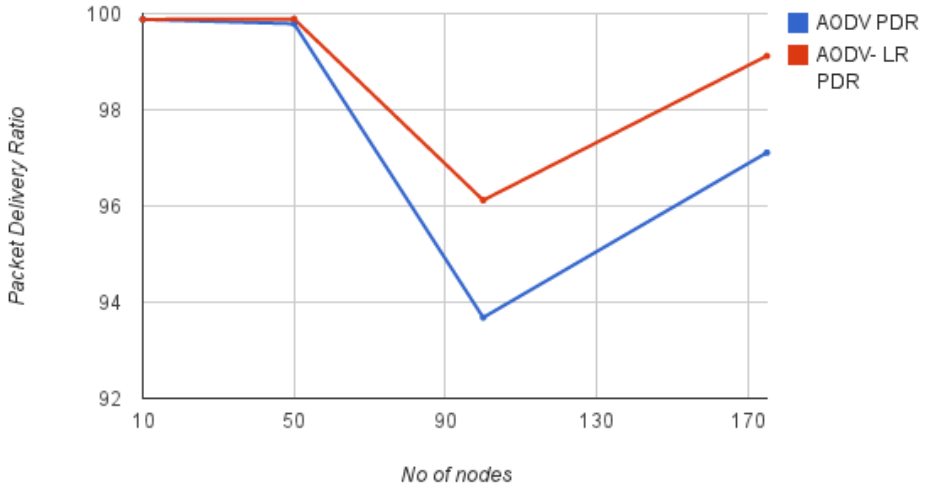

Figure 5.3 Packet Delivery Ratio

In AODV with Local Repair, the Packet Delivery Ratio is expected to increase as the Route repairs by intermediate nodes ensure that paths stay valid and those packets that leave the source reach the destination.

Once again, the number of control packets and the packet delivery ratio may vary significantly from scenario to scenario because of the random nature of our topologies.

Scheme 2- Next-hop Backup based improvement for AODV protocol

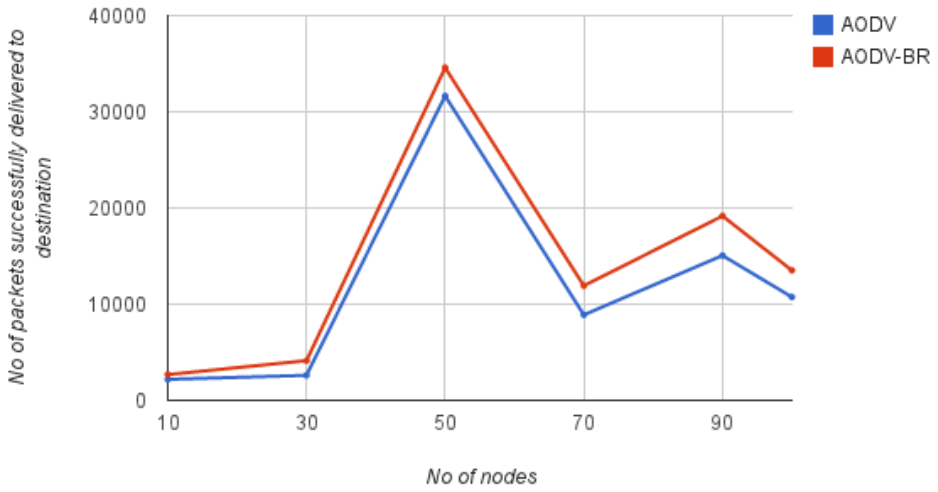

Figure 5.4 Number of packets successfully delivered to destination

AODV with backup route can compensate for link breakages by using the backup node. Once the packet is back on the path, it follows its usual active route. Apart from this, all other parameters are same in AODV and AODV with next-hop backup.

Just like in the Local Repair case, scenario generation in Next-hop Backup Route based improvement is also completely random. These random topologies will react differently in different cases. However, the Backup Route based improvement shows an average improvement of 10 percent when compared to the regular AODV.

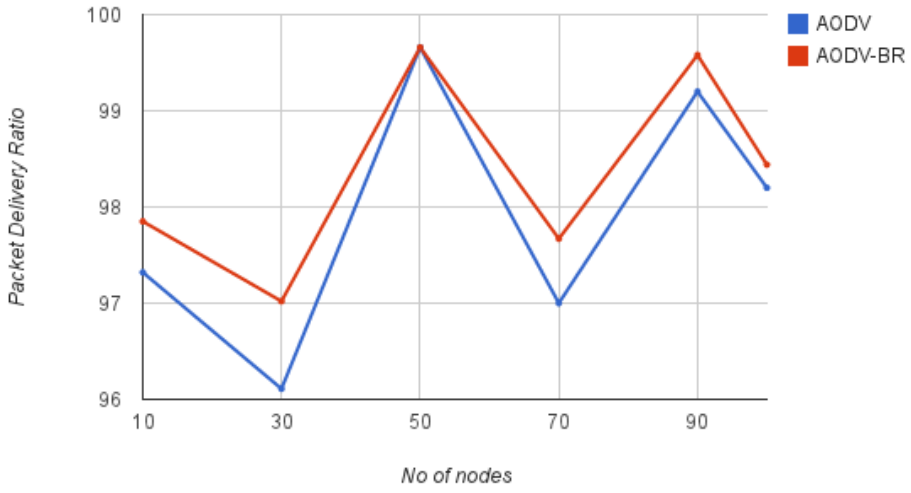

Figure 5.5 Packet Delivery Ratio

In AODV with next-hop backup, when a link breaks, the protocol works out a patch for that link and the remaining part of the link works as it is. Thus, all those packets which are sent are more likely to be delivered than in the regular AODV.

Chapter 6

Conclusion

This thesis describes two different schemes which can be applied to the AODV Routing Protocol. We have shown that both AODV with Local Repair and AODV with Next-hop Backup Route improve the number of packets and the packet delivery ratio.

Both Local Repair and Next-hop Backup Route based improvements make small changes in the operation of AODV. In Local Repair, we force an intermediate node to perform Route Discovery instead of the source node. In Next-hop Backup repair, we designate a backup node for a particular link. Packets are routed through this backup node when the link breaks. Both cases also have their disadvantages, in Local Repair, we are missing out on simpler routes which may be found by the source and instead opting for routes from the intermediate node. In some cases, this might create an unnecessarily long path. However, in most cases Local Repair performs excellently and is definitely an improvement over regular AODV. In the next-hop based backup node repair, we have to generate additional control packets and additional entries in the routing table. This does not necessarily slow down the route generation process because it happens after the active route has been created. Nonetheless, extra control packets are always a burden on the processor.

Several suggestions can be made for future work. Perhaps the most obvious one is a combination of both these schemes. Another improvement would be to develop a protocol such that the nodes nearer to the source undertake Next-hop Backup Route based improvement whereas those closer to the destination can perform Local Repair.

References

[1] E. Schmidt and J. Cohen, “Introduction” in *The New Digital Age: Reshaping the future of people, nations and business,* 1st ed, Knopf, 2013, pp. 3-13

[2] C. E. Perkins, *Ad Hoc Networking,* Illustrated Ed. C.E. Perkins, University of Michigan, Addison-Wesley 2001, pp. 1-28

[3] IEEE Standard for Information Technology- Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11-1997* ,445, 1997

[4] S. Mahan, Self driving car test. Internet: http://www.google.com/about/jobs/lifeatgoogle/self-driving-car-test-steve-mahan.html, [Jun. 15, 2013]

[5] K.P. Valavanis, Unmanned Aerial Vehicles. Springer, 2011

[6] J. Horgan, “Unmanned flight: The drones come home”. National Geographic.[Online]. 223(3). Available: http://ngm.nationalgeographic.com/2013/03/unmanned-flight/horgan-text, [Jun. 15 2013]

[7] R. Bloss, “By air, land and sea, the unmanned vehicles are coming”, Industrial Robot: An International Journal, vol 34, no. 1, pp. 12-16

[8] T. Socolofsky, and C. Kale, A TCP/IP Tutorial, IETF RFC 1180, January 1991; http://tools.ietf.org/pdf/rfc1180.pdf

[9] Information Sciences Institute, University of Southern California, *Intenet Protocol- DARPA Internet Protocol Specification,* IETF RFC 791, September 1981; http://tools.ietf.org/pdf/rfc791.pdf

[10] W.Odom, *Interconnecting Network Devices 1,* Cisco Press, 2013

[11] A. Tannenbaum and D. Wetheral, *Computer Networks,* Prentice Hall, October 2010

[12] W.Odom, *Interconnecting Network Devices 2,* Cisco Press, 2013

[13] R. Deal, *Cisco Certified Network Associate Study Guide,* Mc-Graw Hill Osborne, 2003

[14] C. Hedrick, Routing Information Protocol, IETF RFC 1058, June 1988; http://tools.ietf.org/html/rfc1058

[15] G. Malkin, RIP Version 2, IETF RFC 2453, November 1998; http://tools.ietf.org/html/rfc2453

[16] D. Bertsekas and R Gallager, *Data Networks,* Prentice Hall, 2nd ed, 1992

[17] McQuillan, John M., "The Birth of Link-State Routing,"*Annals of the History of Computing, IEEE* , vol.31, no.1, pp.68,71, Jan.-March 2009

[18] McQuillan, J.M.; Richer, I.; Rosen, E., "The New Routing Algorithm for the ARPANET,"*Communications, IEEE Transactions on* , vol.28, no.5, pp.711,719, May 1980

[19] C. Perkins and P. Bhagwat, Highly Dynamic Destination-Sequenced Distance-Vector (DSDV) Routing for Mobile Computers,*Proc. ACM SIGCOMM Symp. Comm., Architectures and*

Protocols, pp. 234-244, Sept. 1994

[20] Schwartz, M.; Stern, T.E., "Routing Techniques Used in Computer Communication Networks,"*Communications, IEEE Transactions on*, vol.28, no.4, pp.539,552, Apr 1980

[21] P. Patnaik and R. Mall, *Fundamentals of Mobile Computing,* Prentice Hall of India, 2010

[22] Gandhi, S.; Chaubey, N.; Tada, N.; Trivedi, S., "Scenario-based performance comparison of reactive, proactive & Hybrid protocols in MANET," *Computer Communication and Informatics (ICCCI), 2012 International Conference on* , vol., no., pp.1,5, 10-12 Jan. 2012

[23] Morshed, M.M.; Ko, F.I.S.; Dongwook Lim; Rahman, M.H.; Mazumder, M.R.R.; Ghosh, J., "Performance evaluation of DSDV and AODV routing protocols in Mobile ad hoc Networks," *New Trends in Information Science and Service Science (NISS), 2010 4th International Conference on* , vol., no., pp.399,403, 11-13 May 2010

[24] Rahman, J.; Hasan, M.A.M.; Islam, M.K.B., "Comparative analysis the performance of AODV, DSDV and DSR routing protocols in wireless sensor network,"*Electrical & Computer Engineering (ICECE), 2012 7th International Conference on* , vol., no., pp.283,286, 20-22 Dec. 2012

[25] Johnson, D.; Maltz, D.; Broch, J.; DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks, Mobile Computing, edited by Tomasz Emilienski and Hank Korth, Kluwer Academic Publishers, 1996

[26] D. Johnson, D. Maltz and Y. Hu, The Dynamic Source Routing protocol for mobile ad hoc networks for IP v4, IETF RFC 4728, February 2007; http://datatracker.ietf.org/doc/rfc4728/

[27] Karthikeyan, N.; Bharathi, B.; Karthik, S., "Performance analysis of the impact of broadcast mechanisms in AODV, DSR and DSDV," *Pattern Recognition, Informatics and Medical Engineering (PRIME), 2013 International Conference on,* vol., no., pp.144,151, 21-22 Feb. 2013

[28] Sharma, N.; Rana, S.; Sharma, R.M., "Provisioning of Quality of Service in MANETs performance analysis & comparison (AODV and DSR)," *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on* , vol.7, no., pp.V7-243,V7-248, 16-18 April 2010

[29] Perkins, C.E.; Royer, E.M., "ad hoc on-demand distance vector routing," *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on*, vol., no., pp.90,100, 25-26 Feb 1999

[30] C. Perkins., E Royer.; and S. Das, ad hoc On-Demand Distance Vector Routing. IETF RFC 3561, http://www.ietf.org/rfc/rfc3561.txt

[31] Kanthe, Ashok M.; Simunic, Dina; Prasad, Ramjee, "Comparison of AODV and DSR on-demand routing protocols in mobile ad hoc networks," *Emerging Technology Trends in Electronics, Communication and Networking (ET2ECN), 2012 1st International Conference on* , vol., no., pp.1,5, 19-21 Dec. 2012

[32] Bouhorma, M.; Bentaouit, H.; Boudhir, A., "Performance comparison of ad hoc routing protocols AODV and DSR,"*Multimedia Computing and Systems, 2009. ICMCS '09. International Conference on,* vol., no., pp.511,514, 2-4 April 2009

[33] Tomar, G.S.; Sharma, T.; Bhattacharyya, D.; Tai-hoon Kim, "Performance Comparision of AODV, DSR and DSDV under Various Network Conditions: A Survey," *Ubiquitous Computing and Multimedia Applications (UCMA), 2011 International Conference on* , vol., no., pp.3,7, 13-15 April 2011

[34] Marina, M.K.; Das, S.R., "On-demand multipath distance vector routing in ad hoc networks," *Network Protocols, 2001. Ninth International Conference on* , vol., no., pp.14,23, 11-14 Nov. 2001

[35] Royer, E.M.; Melliar-Smith, P.M.; Moser, L.E., "An analysis of the optimum node density for ad hoc mobile networks,"*Communications, 2001. ICC 2001. IEEE International Conference on* , vol.3, no., pp.857,861 vol.3, 2001

[36] Rehmani, M.; Doria, S., and Senouci, M., A tutorial on the implementaiton of ad hoc On Demand Distance Vector (AODV) Protocol in Network Simulator (NS-2), Internet. http://arxiv.org/pdf/1007.4065.pdf [Jun 21, 2013]

[37] Singh, J.; Singh, P.; Rani, S., "Enhanced Local Repair AODV (ELRAODV)," *Advances in Computing, Control, & Telecommunication Technologies, 2009. ACT '09. International Conference on* , vol., no., pp.787,791, 28-29 Dec. 2009

[38] Naidu, P.P.; Chawla, M., "Enhanced ad hoc on demand Distance Vector Local Repair Trial for MANET," *Information and Communication Technologies (WICT), 2011 World Congress on* , vol., no., pp.212,216, 11-14 Dec. 2011

[39] Lu Ding; Li Wan, "Improvement Suggestions to the AODV Routing Protocol," *Wireless Networks and Information Systems, 2009. WNIS '09. International Conference on* , vol., no., pp.370,372, 28-29 Dec. 2009

[40] Lee, S., Belding-Royer, E., and Perkins, C., “Scalability study of the ad hoc on demand distance vector routing protocol” *International Journal of Network Management,* vol.13, no.,pp. 97-114. 2003

[41] Pan, M.; Sheng-Yan Chuang; Sheng-De Wang, "Local repair mechanisms for on-demand routing in mobile ad hoc networks," *Dependable Computing, 2005. Proceedings. 11th Pacific Rim International Symposium on* , vol., no., pp.8 pp.,, 12-14 Dec. 2005

Appendix

Example of a TCL file: tcl file for 10 nodes

```
set val(chan)          Channel/WirelessChannel    ;# channel type

set val(prop)          Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)         Phy/WirelessPhy            ;# network interface type
set val(mac)           Mac/802_11                 ;# MAC type
set val(ifq)           Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)            LL                         ;# link layer type
set val(ant)           Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)        50                         ;# max packet in ifq
set val(nn)            10                         ;# number of mobilenodes
set val(rp)            AODV                       ;# routing protocol
set val(x)             1000                                ;
set val(y)             1000                                ;
set val(simtime)       200.0                               ; #sim time
set val(sc)                        scen10 ;
#set val(em)       EnergyModel ;
#set val(ie)       200.0                       ;

set ftp1start          1;
set ftp2start          10;
set ftpend180;


#
# Initialize Global Variables
#
set ns_     [new Simulator]
set tracefd     [open 10.tr w]
$ns_ trace-all $tracefd

set namtrace [open 10.nam w]          ;# for nam tracing
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
#$ns_ use-newtrace

# set up topography object
set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)

#
# Create God
#

set god_ [ create-god $val(nn) ]


$val(mac) set bandwidth_ 1.0e6

$ns_ node-config -adhocRouting $val(rp) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
                 -channelType $val(chan) \
```

```
             -topoInstance $topo \
             -agentTrace ON \
             -routerTrace ON \
             -macTrace OFF \
             -movementTrace OFF
             #-energyModel $val(em) \
            # -initialEnergy $val(ie) \
            # -batteryModel RTBattery

for {set i 0} {$i < $val(nn) } {incr i} {
        set node_($i) [$ns_ node]
        $node_($i) random-motion 0              ;# disable random motion
}



source $val(sc)



# Define node initial position in nam

for {set i 0} {$i < $val(nn)} {incr i} {

   $ns_ initial_node_pos $node_($i) 20
}



Agent/TCP set packetSize_ 512 ;

set agent1 [new Agent/TCP]
set app1 [new Application/Traffic/CBR]
set sink1 [new Agent/TCPSink]
$app1 set packet_size_ 1024;
$app1 set rate_ 512Kb ;

set agent2 [new Agent/TCP]
set app2 [new Application/Traffic/CBR]
set sink2 [new Agent/TCPSink]
$app2 set packet_size_ 1024;
$app2 set rate_ 512Kb ;

$app1 attach-agent $agent1

$ns_ attach-agent $node_(0) $agent1
$ns_ attach-agent $node_(6) $sink1
$ns_ connect $agent1 $sink1

$app2 attach-agent $agent2

$ns_ attach-agent $node_(7) $agent2
$ns_ attach-agent $node_(9) $sink2
$ns_ connect $agent2 $sink2



# 30 seconds of warmup time for routing
$ns_ at $ftp1start "$app1 start"
#$ns_ at $ftp2start "$app2 start"
```

```
#
# Tell nodes when the simulation ends
#
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $val(simtime) "$node_($i) reset";
}
$ns_ at $val(simtime) "stop"
$ns_ at $val(simtime).01 "puts \"NS EXITING...\" ; $ns_ halt"
proc stop {} {
    global ns_ tracefd
    $ns_ flush-trace
    close $tracefd
}


puts "Starting Simulation..."
$ns_ run
```

AWK file used to calculate end to end delay

```
BEGIN {
    seqno = -1;
   droppedPackets = 0;
   receivedPackets = 0;
    count = 0;
}
{
    if($4 == "AGT" && $1 == "s" && seqno < $6) {
         seqno = $6;
    }
          else if(($4 == "AGT") && ($1 == "r")) {
         receivedPackets++;
  } else if ($1 == "D" && $7 == "tcp" && $8 > 512){
         droppedPackets++;
  }
    #end-to-end delay
    if($4 == "AGT" && $1 == "s") {
         start_time[$6] = $2;
    } else if(($7 == "tcp") && ($1 == "r")) {
       end_time[$6] = $2;
    } else if($1 == "D" && $7 == "tcp") {
         end_time[$6] = -1;
    }
}

END {
    for(i=0; i<=seqno; i++) {
         if(end_time[i] > 0) {
```

```
            delay[i] = end_time[i] - start_time[i];
                count++;
        }
            else
            {
                delay[i] = -1;
            }
    }
    for(i=0; i<=seqno; i++) {
          if(delay[i] > 0) {
              n_to_n_delay = n_to_n_delay + delay[i];
        }
    }
   n_to_n_delay = n_to_n_delay/count;


    print "\n";
   print "GeneratedPackets            = " seqno+1;
    print "ReceivedPackets             = " receivedPackets;
   print "Packet Delivery Ratio       = " receivedPackets/(seqno+1)*100
#"%";
    print "Total Dropped Packets = " droppedPackets;
    print "Average End-to-End Delay    = " n_to_n_delay * 1000 " ms";
    print "\n";
}
```