ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES
FACULTY OF TECHNOLOGY

EVALUATION OF VARIOUS ROUTING PROTOCOLS FOR MOBILE AD-HOC NETWORKS (MANETs)

BY

ETSEGENET AWASH

SEPTEMBER, 2004

EVALUATION OF VARIOUS ROUTING PROTOCOLS FOR MOBILE AD-HOC NETWORKS (MANETs)

By

Etsegenet Awash

A thesis submitted to the school of Graduate Studies of Addis Ababa University in partial fulfillment for the Degree of Master of Science in Computer Engineering

Addis Ababa

September, 2004

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES
FACULTY OF TECHNOLOGY

EVALUATION OF VARIOUS ROUTING PROTOCOLS FOR MOBILE AD-HOC NETWORKS (MANETs)

BY

ETSEGENET AWASH

Approval by Board of Examiners

Dr.-Ing Mohammed Abdo	_____
Chairman, Department of Electrical and Computer Engineering	Signature
Ato Yacob Astatke	_____
Advisor	Signature
Dr. Raimond Kumudha	_____
Examiner	Signature
Prof. Chandra	_____
External Examiner	Signature

Acknowledgement

First and foremost I offer my thanks to GOD, who made this work possible.

I am very grateful to Mekele University for its sponsorship throughout my study.

I express my deepest gratitude to my advisor Ato Yacob Astatke, for his advice, concern, encouragement, and formal guidance regardless of the geographical distance that we have.

My special thanks also go to Dr. Kumudha Raimond, for her guidance and encouragement.

I have to express my sincere appreciation to the staff of Electrical and Computer Engineering, for their encouragement and concern. My special thanks go to Ato Abyot Asalefew for his encouragement, sharing his office and for scarifying his precious time whenever I need help.

Finally, my heartfelt appreciation and indebtedness goes to my family for their love, support, patience and encouragement throughout my study.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

MANET	Mobile Ad-hoc Networks
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
LAN	Local Area Network
DSDV	Destination Sequenced Distance Vector
OLSR	Optimized Link State Routing
DSR	Dynamic Source Routing
AODV	Ad-hoc On-demand Distance Vector
TORA	Temporally Ordered Routing Algorithm
ZRP	Zone Routing Protocol
BRP	BorderCasting Resolution Protocol
IERP	IntErzone Routing Protocol
IARP	IntrAzone Routing Protocol
GSR	Global State Routing
FSR	Fisheye State Routing
CBRP	Cluster Based Routing Protocols
CGSR	Cluster Gateway Switch Routing
DVR	Distance Vector Routing
LSR	Link State Routing
NS-2	Network Simulator Version 2
NAM	Network Animator
CBR	Constant Bit Rate
UDP	User Datagram Protocol

Abstract

Mobile Ad-hoc NETwork (MANET), which is a concept of mobile communication without a fixed communication infrastructure, is one of the current emerging technologies gaining much attention from both researchers and users. Because there is no as such a fixed responsible device router, each computing node is to act as a router and should be willing to forward other's packets.

Routing in mobile computing is a difficult task as we have a very dynamic network topology. Another problem is, the nodes in ad-hoc network are usually laptops and personal digital assistants and are often very limited in resources such as CPU capacity, storage capacity, battery power and bandwidth. This means, any routing protocol running on a mobile node should try to minimize routing or control packets and also be reactive.

A research work group established by IETF, has set a number of requirements to be met by an ad-hoc routing protocol. According to these requirements, a number of routing protocols have been proposed broadly into two main categories, proactive and reactive.

In this thesis, a detailed discussion of routing protocols from both proactive and reactive group is presented followed by simulation work using a discrete event Network Simulator called NS-2. Because these groups of routing protocols fail to satisfy all the requirements of MANET under all conditions such as high mobility, big network size and large number of traffic sources, a new scheme of hybrid routing protocol is proposed. This new routing protocol, called ZRP, is to

take advantages from both proactive and reactive group of protocols. It alleviates scalability problem of other routing protocols by classifying the network into manageable sizes called zones.

ZRP combines two different routing schemes in one protocol. IntrAzone routing uses a proactive protocol to maintain up-to-date routing information to all nodes within its routing zone. By contrast intErzone route discovery is based on a reactive route request and reply scheme. Simulation results of this thesis work shows that ZRP performs well under all conditions especially when we have large network size.

1. Introduction

With the advent of new technologies and the demand for flexibility and ease in working environment, the use of mobile wireless computing is growing fast. Besides their use, mobile wireless networks are assumed to grow in size too. They can function in independent groups, containing some tens of nodes up to several hundreds of nodes. As the network size increases, it becomes common for the nodes to be dispersed in areas which are larger than the radio range of individual nodes. Under such conditions, one has to employ routing techniques such that the out of range nodes may communicate with each other via intermediate nodes. Routing in mobile ad-hoc networks is very difficult as we don't have a fixed infrastructure, the topology is in frequent change, and bandwidth is limited. With a lot of constraints in mobile ad-hoc network, a number of protocols have been proposed so that they can meet these needs of wireless communication. My focus of study is therefore to discuss and analyze these routing protocols based on some performance metrics defined by IETF research group.

1.1. Background

Increased use of laptop computers and an increase in people's mobility have fuelled the demand for wireless networks. The term wireless networking refers to a technology that enables two or more computers to communicate using standard network protocols through radio or infrared signals. The technology was slow, expensive and reserved for mobile situations or hostile environments where cabling was impractical or impossible. With the maturing of industry standards and the deployment of lightweight wireless networking hardware across a broad market section, wireless technology has come of age.

There are two distinct approaches for enabling wireless communication between two hosts. The first approach is to let the existing cellular infrastructure carry data as well as voice. One of the major problems include the problem of handoff, which tries to handle the situation when the connection should be smoothly handed over from one base station to another base station without noticeable delay and packet loss. Another problem is that networks based on the cellular infrastructure are limited to places where there exists such a cellular infrastructure.

The second approach is to form an ad-hoc network among all users wanting to communicate with each other. This means that all users participating in the ad-hoc network must be able to forward data packets to make sure that the packets are delivered from source to destination. This form of networking is limited in range by the individual nodes' transmission ranges and is typically smaller compared to the range of cellular systems. This does not mean that the cellular approach is better than the ad-hoc approach. Ad-hoc networks have several advantages compared to traditional cellular systems. These include:

- On demand setup
- Fault tolerance
- Unconstrained connectivity

Ad-hoc networks do not rely on any pre-established infrastructure and therefore can be deployed in places with no infrastructure. This is useful in disaster recovery situations, and places with non-existing or damaged communication infrastructure where rapid deployment of a communication network is needed. Ad-hoc networks can also be useful in conferences where people participating in the conference can form a temporary network without engaging the services of any pre-existing network.

1.2. Problem Description

Mobile Ad Hoc Networks (MANETs) consist of a group of mobile nodes, which form communication network without prior infrastructure. Each node in the network is therefore responsible for providing services to other nodes in order to realize the ad hoc communication capability. Because of the lack of a common access point, route information in MANETs is very much unstable and hence makes routing in MANET complicated.

In a wired network, if node A is for example reachable from node B, then it is obvious that node B is reachable from node A. But this may not be the case in MANETs. In the wireless medium, because of radio transmission range limitations, each and every node may not have enough power to the reach the destination node. Therefore, communication between two remote hosts needs to be established through a multi-hop link. To accomplish this task, various routing protocols capable of adapting to changing network topologies have been

proposed. Those routing protocols should minimize the usage of valuable resources such as bandwidth, CPU, and power.

In order to address the above problems, a lot of MANET routing protocols have been proposed. In addition, an Internet Engineering Task Force has established a MANET working Group, whose task is to standardize and evaluate these protocols based on some performance measuring criteria.

By understanding the working principles and pros and cons of the existing routing protocols, it is possible to bring a new efficient routing protocol for MANETs. Therefore, performance evaluation of various existing MANET routing protocols is crucial to the design of an efficient new routing protocol. Evaluating these routing protocols based on some metrics will also help to standardize them.

1.3. Related work

A number of routing protocols are proposed for mobile ad-hoc networks [1][2][3][4][5][6][7][8]. And it is known that all the proposed protocols are not efficient and do not qualify all the desired properties of Ad-hoc routing protocols stated below in this thesis.

Despite of the various protocols, few comparisons between the different protocols have been made. Of the work that has done in this field, the work done by the Monarch Project at Carnegie Mellon University (CMU)[20] has compared some of the different proposed routing protocols and evaluated them based on some quantitative matrices.

There are also several other simulation results of these routing protocols that have been conducted using various simulation software packages. However, they are still not quite sufficient to make decisions as to which routing protocols are better for certain applications. That is the reason a lot of research is still being conducted in the design and implementation of new routing protocols for MANETs.

1.4. Methodology

The methodology that will be used to conduct this research work are listed below

1. **Literature survey**

This is assessing any published and unpublished work to gather any information about the study area relevant to the research work. This involves research in libraries, and the internet. The next two chapters describe the literature survey that is made for this thesis.

2. **Software simulation**

I have used the discrete event NS-2 network simulator for simulation, analysis and comparison of the ad-hoc routing protocols. I have tried to discuss the theoretical background of each routing protocol and how well it represents real life by simulating each routing protocol using a scenario, which resembles to real life cases. Three key performance metrics that I have measured for each routing protocols are:

- Throughput : ratio of the packets delivered to the total numbers of packets sent
- Average end to end delay: time taken for the packets to reach the destination
- Routing Overhead: the number of control packets for the routing protocol over the number of packets of data packets sent.

I have compared them using these performance measures, and included my recommendations.

3. Implementation of proposed routing protocols

Since NS-2 network simulator does not support all the proposed protocols, I have to implement some protocols on NS-2 for simulation purposes.

2. Wireless Networks

2.1. Introduction

Wireless network is an emerging new technology that will allow users to access information and services electronically, regardless of their geographic position. The term wireless networking refers to a technology that enables two or more computers to communicate using standard network protocols through radio or infrared signals. The technology was slow, expensive and reserved for mobile situations or hostile environments where cabling was impractical or impossible. With the maturing of industry standards and the deployment of lightweight wireless networking hardware across a broad market section, wireless technology has come of age.

The industry standard, which governs wireless networking, is 802.11, which is produced by the Institute of Electrical and Electronic Engineers (IEEE). Wireless networking hardware requires the use of underlying technology that deals with radio frequencies as well as data transmission. 802.11 is a standard defining all aspects of Radio Frequency for Wireless networking.

In IEEE's proposed standard for wireless LANs (IEEE 802.11), there are two different ways to configure a network: ad-hoc and infrastructure.

Network with infrastructure consists of fixed and wired gateways. A mobile node communicates with a bridge in the network (called base station) within its communication radius. The mobile unit can move geographically while it is communicating. When it goes out of range of one base station, it connects with new base station and starts communicating through it. This is called handoff. This type of wireless network uses an access point (base station), which acts as a hub, providing connectivity for the wireless computers. It can connect (or "bridge") the wireless LAN to a wired LAN, allowing wireless computer access to LAN resources, such as file servers or existing
Internet Connectivity.

There are two types of access points:

- Dedicated hardware access points offer comprehensive support of most wireless features, but check the requirements carefully.
- Software Access Points which run on a computer equipped with a wireless network interface card as used in an ad-hoc or peer-to-peer wireless network

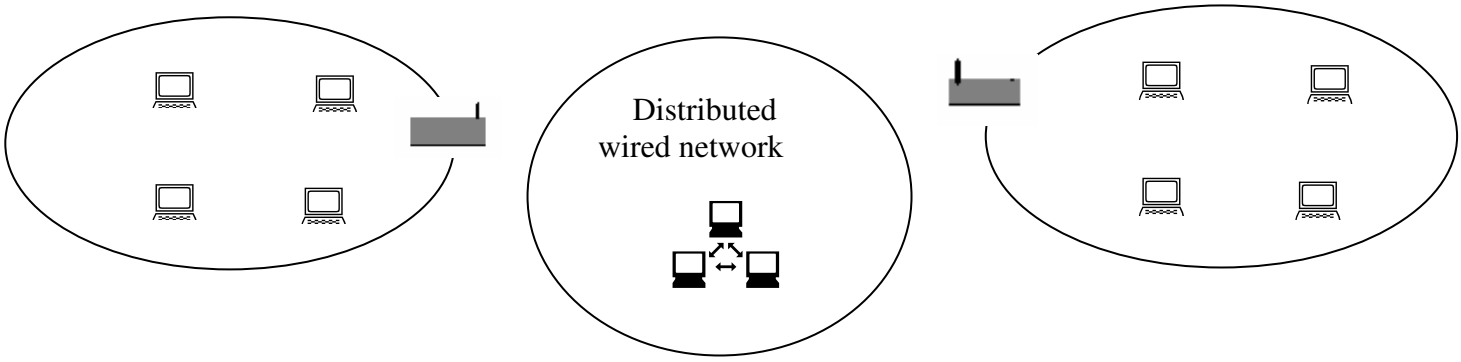


Fig 2.1 Infrastructure Network with two access points

In contrast to infrastructure-based networks, in ad hoc networks all nodes are mobile and can be connected dynamically in an arbitrary manner. All nodes of these networks behave as routers and take part in discovery and maintenance of routes to other nodes in the network. More formally, an ad hoc network is a collection of wireless nodes (see figure2.2) that can be rapidly deployed as a multi-hop packet radio network without the aid of any established infrastructure or centralized administration and without any user-initiated configuration actions.

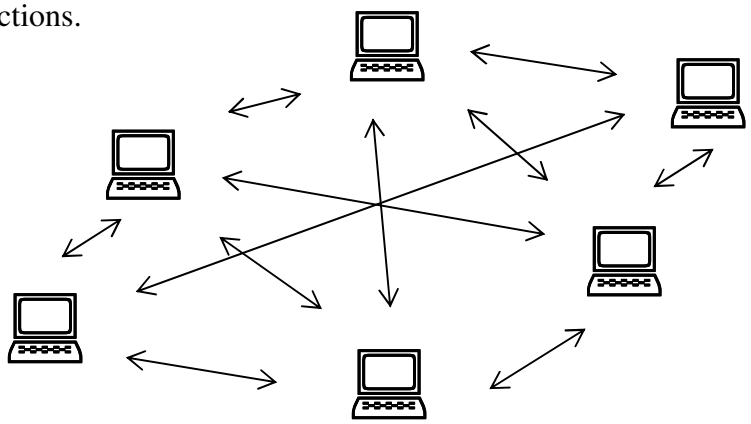


Fig 2.2 Ad-hoc or peer-to-peer networking

2.2. Mobile Ad-hoc Networks

A Mobile Ad-hoc Network (MANET) is a temporary wireless network composed of mobile nodes, in which an infrastructure is absent. There are no dedicated routers, servers, access points and cables. Because of its speedy and convenient deployment, robustness, and low cost, a MANET can find its applications in the following areas:

- Military use (e.g. a network in the battlefield)
- Search and rescue
- Vehicle-to-vehicle communication in intelligent transportation
- Temporary networks in meeting rooms, airports, etc.
- Personal Area Networks connecting cell phones, laptops, smart watches, and other wearable computers.

If two mobile nodes are within each other's transmission range, they can communicate with each other directly; otherwise, the nodes in between have to forward the packets for them. In such a case, every mobile node has to function as a router to forward the packets for others.

Traditional routing protocols used in hardwired networks, such as distance vector protocols (e.g. RIP) and link state protocols (e.g., OSPF) cannot be applied in the MANET directly for the following reasons:

1. There may be uni-directional links between nodes;
2. There are more than one eligible path between two nodes;
3. The consumption of bandwidth and power supply incurred by periodic routing information updates is considerable;
4. The routing fabrics converge slowly in contrast to rapid topology change.

2.3. Characteristics of Ad-hoc Networks

An ad hoc network is a collection of mobile nodes forming a temporary network without the aid of any centralized administration or standard support services regularly available on conventional networks.

The MANET working group has defined some unique properties of ad hoc networks in RFC 2501 [10]. The properties does not directly relate to performance. However, they describe the very nature of ad hoc networks and in that sense they formulate the boundary conditions to ad hoc networking. In a manner they impact on performance, since they greatly affect on the design of ad hoc routing protocols.

The following characteristics are defined by the MANET working group in RFC 2501:

1. **Dynamic topologies**

   The topic refers to the most essential property of an ad hoc network: Nodes can move arbitrarily with respect to other nodes in the network.

2. **Bandwidth-constrained**

   Nodes in an ad hoc network are mobile. Thus, they are using radio links that have far lower capacity than hardwired links could use. In practice the realized throughput of a wireless network is less than a radio's theoretical maximum transmission rate.

3. **Energy constrained operation**

   Mobile nodes are likely to rely on batteries. That is why the primary design criteria may sometimes be energy conservation.

4. **Limited physical security**

   In general, radio networks are vulnerable to physical security threats compared to fixed networks. The possibility of eavesdropping, spoofing and DoS attacks is higher. Existing link security techniques can be applied.

However, a single point failure in an ad hoc network is not as crucial as in more to that of centralized networks.

2.4. Routing

Routing is the process that a node uses to forward packets toward the destination network. A node makes decisions based upon the destination IP address of a packet. All devices along the way use the destination IP address to point the packet in the correct direction so that the packet eventually arrives at its destination. In order to make the correct decisions, nodes must learn the direction to remote networks. When nodes use dynamic routing, this information is learned from other nodes by exchanging route information using routing protocols. When static routing is used, a network administrator configures information about remote networks manually.

2.4.1. Routing protocols

A routing protocol is a way of exchanging route information used between nodes. Routing protocol allows one node to share information with other nodes regarding the networks it knows about as well as its proximity to other routers. The information a node gets from another node, using a routing protocol, is used to build and maintain a routing table so that the node can make decisions on forwarding a packet to its destination.

Those routing algorithms used for wired existing networks we call them as conventional routing protocols. They can be classified as Distance vector or Link state routing algorithms. These algorithms are good and well tested on networks which have some how semi-static topology that is with low mobility.

2.4.2. Classification

There are different criteria for designing and classifying routing protocols for wired and wireless ad hoc networks. For example, what routing information is exchanged; when and how the routing information is exchanged, when and how routes are computed and so on? We will discuss these criteria in this section.

There are different criteria for designing and classifying routing protocols for wired and wireless ad hoc networks. For example, what routing information is exchanged; when and how the routing information is exchanged, when and how routes are computed and so on. We will discuss these criteria in this section.

2.4.2.1. Link state routing vs. Distance vector routing

As with conventional wired networks, Link state routing (LSR) and distance vector routing (DVR) are two underlying mechanisms for routing in wireless ad hoc networks. In LSR [11], routing information is exchanged in the form of link state packets (LSP). The LSP of a node includes link information about its neighbors. Any link change will cause LSPs to be flooded into the entire network immediately. Every node can construct and maintain a global network topology from the LSPs it receives, and compute, by itself, routes to all other nodes. The problem with LSR is that excessive routing overhead may be incurred because nodes in a wireless ad hoc network move quickly and the network topology changes fast.

In DVR [11], every node maintains a distance vector, which includes, but is not limited to, the triad (destination ID, next hop, (shortest) distance) for every destination. Every node periodically exchanges distance vectors with its neighbors. When a node receives distance vectors from its neighbors, it computes new routes and updates its distance vector. The complete route from a source to a destination is formed, in a distributed manner, by combining the next hop of nodes on the path from the source to the destination. The problems with DVR are slow convergence and the tendency of creating routing loops.

2.4.2.2. Pre-computed routing vs. on-demand routing

Depending on when the route is computed, routing protocols can be divided into two categories: pre-computed routing and on-demand routing.

Pre-computed routing is also called proactive routing or table-driven routing. In this method, the routes to all destinations are computed a priori. In order to compute routes in advance, nodes need to store the entire or partial information about link states and network topology. In order to keep the information up to date, nodes need to update their information periodically or whenever the link state or network topology changes. The advantage of pre-computed routing is that when a source needs to send packets to a destination, the route is already available, i.e., there is no latency. The disadvantage is that some routes may never be used. Another problem is that the dissemination of routing information will consume a lot of the scarce wireless network bandwidth when the link

state and network topology change fast (this is especially true in a wireless ad hoc network). The conventional LSR and DVR are examples of proactive routing.

On-demand routing is also called reactive routing. In this method, the route to a destination may not exist in advance and it is computed only when the route is needed. The idea is as follows: When a source needs to send packets to a destination, it first finds a route or several routes to the destination. This process is called route discovery. After the route(s) are discovered, the source transmits packets along the route(s). During the transmission of packets, the route may be broken because the node(s) on the route move away or go down. The broken route needs to be rebuilt. The process of detecting route breakage and rebuilding the route is called route maintenance. The major advantage of on-demand routing is that the precious bandwidth of wireless ad hoc networks is greatly saved [9] because it limits the amount of bandwidth consumed in the exchange of routing information by maintaining routes to only those destinations to which the routers need to forward data traffic. On-demand routing also obviates the need for disseminating routing information periodically, or flooding such information whenever a link state changes. The primary problem with on-demand routing is the large latency at the beginning of the transmission caused by route discovery.

Apart from proactive route computation and reactive route discovery, there is another routing mechanism, called flooding [9]. In flooding, no route will be computed or discovered. A packet is broadcast to all nodes in a network with the expectation that at least one copy of the packet will reach the destination. Scoping [9] may be used to limit the overhead of flooding. Flooding is the easiest routing method because it requires no knowledge of the network topology. Under light traffic conditions, flooding can be reasonably robust. However, it generates an excessive amount of traffic in heavy traffic or in a large network and it is difficult to achieve flooding reliably when the topology is highly dynamic. Flooding is generally used to transmit control packets (e.g., routing information), not data packets.

2.4.2.3. Periodical update vs. event-driven up-date

Routing information needs to be disseminated to network nodes in order to ensure that the knowledge of link state and network topology remains up-to-date. Based on when the routing information will be disseminated, we can classify routing protocols as periodical update and event-driven update protocols. Periodical update protocols disseminate routing information periodically. Periodical updates will simplify protocols and maintain network stability, and most importantly, enable (new) nodes to learn about the topology and the state of the network. However if the period between updates is large, the protocol may not keep the information up-to-date. On the other hand, if the period is small, too many routing packets will be disseminated which consumes the precious bandwidth of a wireless network. In an event-driven update protocol, when events occur, (such as when a link fails or a new link appears), an update packet will be broadcast and the up-to-date status can be disseminated over the network soon. The problem might be that if the topology of networks changes rapidly, a lot of update packets will be generated and disseminated over the network which will use a lot of precious bandwidth, and furthermore, may cause too much fluctuation of routes. Periodical update and event-driven update mechanisms can be used together, forming what is called a hybrid update mechanism. For example, in DSDV [2], a node broadcasts its distance-vector periodically. Moreover, whenever a node finds that a link is broken, it distributes a message immediately.

2.4.2.4. Flat structure vs. hierarchical structure

In a flat structure, all nodes in a network are at the same level and have the same routing functionality. Flat routing is simple and efficient for small networks. The problem is that when a network becomes large, the volume of routing information will be large and it will take a long time for routing information to arrive at remote nodes.

For large networks, hierarchical (cluster-based) routing may be used to solve the above problems [8]. In hierarchical routing, the nodes in the network are dynamically organized into partitions called clusters, and then the clusters are aggregated again into larger partitions called super clusters and so on. Organizing a network into clusters, helps maintain a relatively stable network topology [8], [12]. The high dynamics of membership and network topology is limited within clusters. Only stable and high level information

such as the cluster level or the super cluster level will be propagated across a long distance, thus the control traffic (or routing overhead) may be largely reduced. Within a cluster, the nodes may have complete topology information about its cluster and proactive routing may be used. If the destination is in a different cluster from the source, inter cluster routing must be used. Inter cluster routing is generally reactive, or a combination of proactive and reactive routing as in [12]. Similar to cellular structure in cellular systems, a hierarchical cluster is readily deployable to achieve some kind of resource reuse such as frequency reuse and code reuse [9], [15] and interference can be reduced when using different spreading codes across clusters.

2.4.2.5. Source routing vs. hop-by-hop routing

Some routing protocols place the entire route (i.e., nodes in the route) in the headers of data packets so that the intermediate nodes only forward these packets according to the route in the header. Such a routing is called “source routing”. Source routing has the advantage that intermediate nodes do not need to maintain up-to-date routing information in order to route the packets they forward, since the packets themselves already contain all the routing decisions. This fact, when coupled with on-demand route computation, eliminates the need for the periodic route advertisement and neighbor detection packets required in other kinds of protocols [1]. The biggest problem with source routing is that when the network is large and the route is long, placing the entire route in the header of every packet will waste a lot of scarce bandwidth.

In hop-by-hop routing, the route to a destination is distributed in the “next hop” of the nodes along the route. When a node receives a packet to a destination, it forwards the packet to the next hop corresponding to the destination. The problems are that all nodes need to maintain routing information and there may be a possibility of forming a routing loop.

2.4.2.6. Single path vs. multiple paths

Some routing protocols will find a single route from a source to a destination, which results in simple protocol and saves storage. Other routing protocols will find multiple routes,

which have the advantages of easy recovery from a route failure and being more reliable and robust. Moreover, the source can select the best one among multiple available routes.

Based on the classification of the above concepts of routing protocols, a number of routing protocols are proposed for Mobile Ad-hoc Networks (MANETs). The next chapter contains a detail discussion on the concepts and working principles of these routing protocols.

3. Ad-hoc Routing Protocols

Ad-hoc networks are network architectures that enable rapid deployment. They are able to adapt to the rapidly changing traffic conditions and mobility patterns of the network nodes. The characteristics of ad-hoc networks include the lack of fixed infrastructure, a distributed peer-to-peer mode of operation, multi-hop routing, and relatively frequent change in network topology. An ad-hoc routing protocol has to enable a node that is sending a packet to obtain information on the reachability of its neighboring nodes in a network of dynamic topology. And the basic problem with the conventional routing protocols(Distance vector and Link state) is that are designed for a semi-static topology, which means that they would have problems to converge to steady state in an ad-hoc network with very frequency changing topology.

3.1. Desirable Characteristics

The MANET working group also defines some desirable qualitative properties of ad hoc routing protocols in RFC 2501 [10]. They are useful when assessing performance or suitability of an ad hoc routing protocol thus they are worth to mention in the context of this paper. The following properties are defined in RFC 2501:

Distributed operation

This property is essential to ad hoc networks. It is self-evident that ad hoc networks operate in distributed manner because of their very nature. Therefore, the routing protocols should be distributed, not dependent on a centralized controlling node.

Loop-freedom

This property is generally desirable. It refers to avoiding packets spinning around in the network for arbitrary time. Solutions such as TTL values, sequence numbers can be used to limit performance effects of the problem. However, a more structured or a sophisticated solution will probably lead to better overall performance.

Demand based operation

Ad hoc routing does not have to assume uniform traffic load in a network but it can adapt to traffic patterns on need basis. This will increase route discovery delay but when implemented intelligently bandwidth and energy resources can be more efficiently utilized.

Proactive operation

This is opposite to demand-based operation. If additional delays that occur in demand-based operation are unacceptable, proactive approach can be used especially when energy and bandwidth capacities support the use of proactive operation.

Security

Ad hoc routing protocols are exposed to several kinds of attacks. Maintaining link layer security is in practice harder with ad hoc networks than with fixed networks. Sufficient routing protocols security is desirable. Sufficient within this context covers prohibiting disruption or modification of protocol operation. Probably emerge rarely.

Power Conservation

The nodes in ad hoc network can be laptops and thin clients such as Personal Digital Assistance (PDAs) that are very limited in battery power and therefore uses some sort of stand-by mode to save power. It is therefore important that the routing protocols have support for these sleep-modes.

Unidirectional support

The radio environment can cause the formation of unidirectional links. Utilization of these links and not only the bi-directional links improves the routing performance.

3.2. Routing protocols of Mobile Ad-hoc Networks

A number of routing protocols have been suggested for ad-hoc networks [1], [2], [3], [4], [5], [6]. These protocols can be classified into three main categories: proactive (table-driven), reactive (source-initiated or demand-driven) and Hybrid (taking both proactive and reactive).

Proactive routing protocols as discussed above, attempt to keep an up-to-date topological map of the entire network. With this map, the route is known and immediately available when a packet needs to be sent. Proactive protocols are traditionally classified as either distance-vector or link-state protocols. The former are based on the distributed Bellman-Ford (DBP) algorithm, which is known for slow convergence because of the "counting-to-infinity" problem. To address the problem, the Destination-Sequenced Distance-Vector routing (DSDV) [2] protocol was proposed for ad-hoc networks.

On the other hand, link-state protocols, as represented by OSPF [21], have become standard in wired IP networks. They converge more rapidly, but require significantly more control traffic. Since ad-hoc networks are bandwidth limited and their topology changes often, an Optimized Link-State Protocol (OLSR) [5] has been proposed. While being suitable for small networks, some scalability problems can be seen on larger networks. The need to improve convergence and reduce traffic has led to algorithms that combine features of distance-vector and link-state schemes.

In contrast to proactive routing, reactive routing does not attempt to continuously determine the network connectivity. Instead, a route determination procedure is invoked on demand when a packet needs to be forwarded. The technique relies on queries that are flooded throughout the network.

Reactive route determination is used in the Temporally Ordered Routing Algorithm (TORA) [4], the Dynamic Source Routing (DSR) [1] and the Ad-hoc On-demand Distance Vector (AODV) [3] protocols. In DSR and AODV, a reply is sent back to the query source along the reverse path that the query traveled. The main difference is that DSR performs

source routing with the addresses obtained from the query packet, while AODV uses next-hop information stored in the nodes of the route. In contrast to these protocols, TORA creates directed acyclic graphs rooted at the destination by flooding the route replies in a controlled manner. [4]

The hybrid category is the newest class of routing protocols for ad hoc networks. Likely the most recognized protocol of this type is the Zone Routing Protocol (ZRP). ZRP makes use of proactive routing in order to keep, maintain, and utilize the paths to nearby nodes (nodes within the local zone). Due to their proximity, the paths to these nodes can be kept relatively easily and inexpensively. For more distant nodes, source-initiated on-demand routing is used. Although this requires additional delay in order to discover the route, it eliminates wasted routing effort to these nodes.

IETF has working group named as MANET [10] that is working in the field of ad-hoc networks. They are currently developing routing specification for ad-hoc IP networks that support scale to a couple of hundred nodes. They are also working on routing protocols for mobile ad hoc networks. There are several routing protocols proposed drafts published by IETF as RFC.

Of these proposed protocols, I have chosen to analyze and compare (simulation based) the following routing protocols. These are:

- Destination Sequenced Distance Vector(DSDV) proactive approach
- Optimized Link State Routing (OLSR) proactive approach
- Ad-hoc On-demand Distance Vector (AODV) reactive approach
- Dynamic Source Routing (DSR) reactive approach
- Temporal Ordered Routing Algorithm (TORA) also reactive
- Zone Routing Protocol (ZRP) hybrid approach.

I have also discussed some other routing protocols theoretically for comparison.

3.3. Destination Sequenced Distance Vector (DSDV)

In wired networks routing protocols traditionally use either distance-vector or link-state algorithms. Both of which allow a mobile node to find the next hop to reach a destination via the shortest path. In principle any such protocol can be used in an ad hoc network. The main draw back of both link-state and distance-vector protocols are that they take too long to converge and have a high message complexity. Because of the limited bandwidth of mobile nodes, message complexity must be kept low. In addition to the rapidly changing topology of the network requires that routes be found quickly. The Destination-Sequenced Distance Vector (DSDV) protocol is an adaptation of the Bellman-Ford routing protocol and is specifically targeted for the ad hoc network [2]. DSDV is adapted from the Routing Information Protocol (RIP) used in wired networks to ad hoc network routing.

3.3.1. Routing and Routing Table

Each mobile node maintains a routing table of all the possible destinations within the network, the metric and next hop to each destination and a sequence number generated by the destination node. This allows packets to be transmitted between the nodes of an ad-hoc network to reach the destination. Each mobile node updates the routing table periodically or when significant new information is available.

When a change in the network topology is detected, each mobile node advertises routing information periodically or immediately by broadcasting a routing table update packet. The update packet is sent to its direct neighbors with a metric of one. This indicates that the receiving neighbor is one hop away from the node. After receiving the update packet the neighbor updates their routing table and increments the metric by one and retransmits the update packet to their direct neighbors. This process is repeated until all the nodes in the ad hoc network have received a copy of the update packet with a corresponding metric. The update data is kept for a while to wait for the arrival of the best routing for each destination. If a node receives multiple update packets for the same destination the routes with the most recent sequence numbers are always used for forwarding decisions.

As network topology is dynamic the elements in the routing table of each node needs to be dynamic to reflect these changes. To maintain this consistency route updates need to be frequent and fast enough to ensure that each mobile node can locate all the other mobile nodes in the network.

The use of sequence numbers allows a mobile node to distinguish between new and old routes, there by avoiding the formation of routing loops. Routing table updates are broadcast throughout the network at intervals to maintain constancy. To alleviate possible high bandwidth use during an update two possible types of packet may be used. The first is known as full dump. This carries all available routing information and can be of a considerable size. Incremental dump packets are used to relay only information that has changed since the last full dump. As an incremental packet does not contain all information they generate smaller packets. A mobile node will maintain an additional table that stores information from incremental packet data.

A new route broadcasts information that contains its address, the number of hops to reach the destination, the sequence number of the information received as well and a unique number for that broadcast; the route with the most recent sequence number is used. For routes with the same sequence number, the route with the smallest metric is used.

3.3.2.Unidirectional Links

DSDV assumes that all wireless links are bi-directional, even though it is possible that unidirectional links may exist in an ad hoc network. Unidirectional links can cause a number of problems for DSDV, these being:

- Knowledge Asymmetry: Mobile nodes may know of neighbors but cannot assume that they know of them.
- Node Unreachability: In DSDV the destination node initiates the path updates. Over unidirectional links, there may be no way that a node can broadcast its.

3.4. Optimized Link State Routing Protocol (OLSR)

3.4.1. Protocol Overview

Optimized Link State Protocol (OLSR) is a proactive routing protocol, so the routes are always immediately available when needed. OLSR is an optimization version of a pure link state protocol. So the topological changes cause the flooding of the topological information to all available hosts in the network. To reduce the possible overhead in the network protocol uses Multipoint Relays (MPR). The idea of MPR is to reduce flooding of broadcasts by reducing the same broadcast in some regions in the network, more details about MPR can be found later in this section. Another reduce is to provide the shortest path. The reducing the time interval for the control messages transmission can bring more reactivity to the topological changes. [5]

OLSR uses two kinds of the control messages: Hello and Topology Control (TC). Hello messages are used for finding the information about the link status and the host's neighbors. With the Hello message the Multipoint Relay (MPR) Selector set is constructed which describes which neighbors has chosen this host to act as MPR and from this information the host can calculate its own set of the MPRs. the Hello messages are sent only one hop away but the TC messages are broadcasted throughout the entire network.

TC messages are used for broadcasting information about own advertised neighbors which includes at least the MPR Selector list. The TC messages are broadcasted periodically and only the MPR hosts can forward the TC messages.

There is also Multiple Interface Declaration (MID) messages, which are used for informing other host that the announcing host can have multiple OLSR interface addresses. The MID message is broadcasted throughout the entire network only by MPRs. There is also a "Host and Network Association" (HNA) message which provides the external routing information by giving the possibility for routing to the external addresses. The HNA message provides information about the network- and the netmask addresses, so that OLSR host can consider that the announcing host can act as a gateway to the announcing set of addresses. The HNA is considered as a generalized version of the TC message with only difference that the TC

message can inform about route canceling while HNA message information is removed only after expiration time. For further information of HNA and MID look at [5]

3.4.2.Neighbor Discovery

The link in the ad hoc network can be either unidirectional or bidirectional so the host must know this information about the neighbors. The Hello messages are broadcasted periodically for the neighbor sensing. The Hello messages are only broadcasted one hop away so that they are not forwarded further. When the first host receives the Hello message from the second host, it sets the second host status to asymmetric in the routing table. When the first host sends a Hello message and includes that, it has the link to the second host as asymmetric, the second host set first host status to symmetric in own routing table. Finally, when second host send again Hello message, where the status of the link for the first host is indicated as symmetric, then first host changes the status from asymmetric to symmetric. In the end both hosts knows that their neighbor is alive and the corresponding link is bidirectional.

The Hello messages are used for getting the information about local links and neighbors. The Hello messages periodic broadcasting is used for link sensing, neighbor's detection and MPR selection process. Hello message contains: information how often the host sends Hello messages, willingness of host to act as a Multipoint Relay, and information about its neighbor. Information about the neighbors contains: interface address, link type and neighbor type. The link type indicates that the link is symmetric, asymmetric or simply lost. The neighbor type is just symmetric, MPR or not a neighbor. The MPR type indicates that the link to the neighbor is symmetric and that this host has chosen it as Multipoint Relay. [13], [14]

3.4.3.Multipoint Relays

The Multipoint Relays (MPR) is the key idea behind the OLSR protocol to reduce the information exchange overhead. Instead of pure flooding, the OLSR uses MPR to reduce the number of the host, which broadcasts the information throughout the network. The MPR is a host's one hop neighbor, which may forward its messages. The MPR set of host

is kept small in order for the protocol to be efficient. In OLSR, only the MPRs can forward the data throughout the network. [5]

Each host must have the information about the symmetric one hop and two hop neighbors in order to calculate the optimal MPR set. The Fig.3.1 is taken from [13] shown below is to illustrate these concepts. Information about the neighbors is taken from the Hello messages. The two hop neighbors are found from the Hello message because each Hello message contains all the hosts' neighbors. Selecting the minimum number of the one-hop neighbors, which covers all the two hop neighbors, is the goal of the MPR selection algorithm. Also each host has the Multipoint Relay Selector set, which indicates which hosts has selected the current host to act as a MPR. See more about MPR Selector at [5].

When the host gets a new broadcast message, which is need to be spread throughout the network and the message's sender interface address is in the MPR Selector set, then the host must forward the message. Due to the possible changes in the ad hoc network, the MPR Selectors sets are updated continuously using Hello messages. [5]

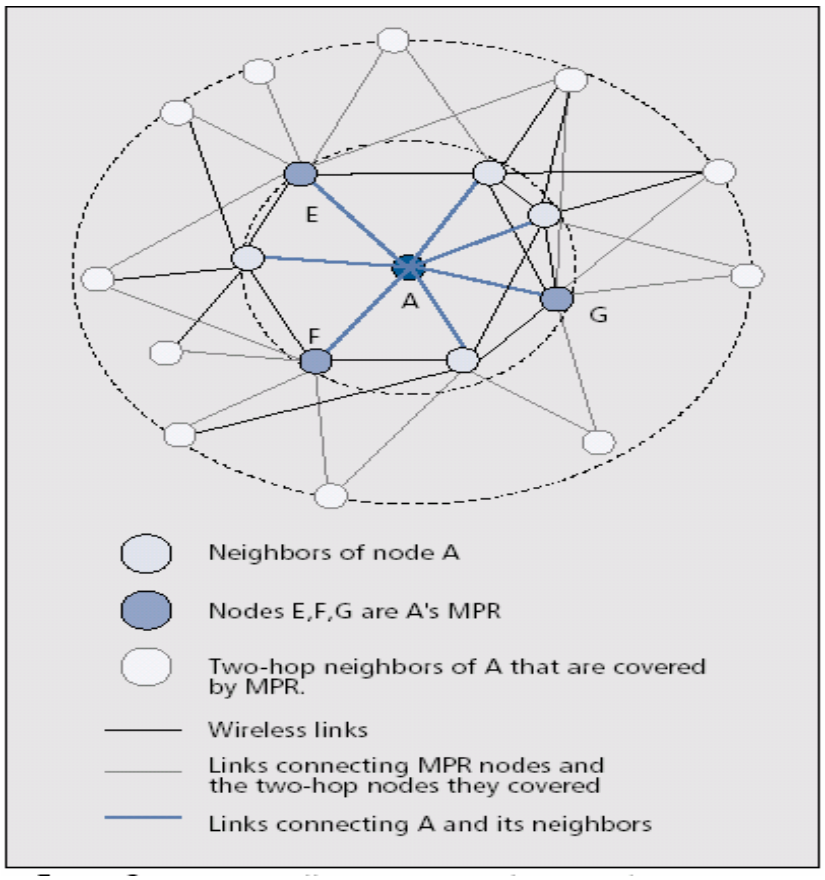


Fig 3.1 OLSR: an illustration of multipoint relays. [13]

3.4.4.Topology Information

In order to exchange the topological information and build the topology information base the host that were selected as MPR need to sent the topology control (TC) message. The TC messages are broadcasted throughout the network and only MPR are allowed to forward TC messages. The TC messages are generated and broadcasted periodically in the network.

The TC message is sent by a host in order to advertise own links in the network. The host must send at least the links of its MPR selector set. The TC message includes the own set of advertised links and the sequence number of each message. The sequence number is used to avoid loops of the messages and for indicating the freshness of the message, so if

the host gets a message with the smaller sequence number it must discard the message without any updates. The host must increment the sequence number when the links are removed from the TC message and it should increment the sequence number when the links are added to the message. [5]

3.4.5.Routing Table

The host maintains the routing table, the routing table entries have following information: destination address, next address, number of hops to the destination and local interface address. Next address indicates the next hop host.

The information is got from the topological set (from the TC messages) and from the local link information base (from the Hello messages). So if any changes occur in these sets, then the routing table is recalculated. Because this is proactive protocol then the routing table must have routes for all available hosts in the network. The information about broken links or partially known links is not stored in the routing table. [5]

The routing table is changed if the changes occur in the following cases: neighbor link appear or disappear, two hops neighbor is created or removed, topological link is appeared or lost or when the multiple interface association information changes. But the update of this information does not lead to the sending of the messages into the network. For finding the routes for the routing table entry, the shortest path algorithm is used. [5]

3.5. Dynamic Source Routing Protocol (DSR)

Dynamic Source Routing (DSR) [1] is designed to allow nodes to dynamically discover a source route across multiple network hops to any destination in the ad hoc network. When using source routing, each packet to be routed carries in its header the complete, ordered list of nodes through which the packet must pass. A key advantage of source routing is that intermediate hops do not need to maintain routing information in order to route the packets they receive, since the packets themselves already contain all necessary routing information. An example of a packet moving through an ad hoc network with source routing is illustrated in the following figure.

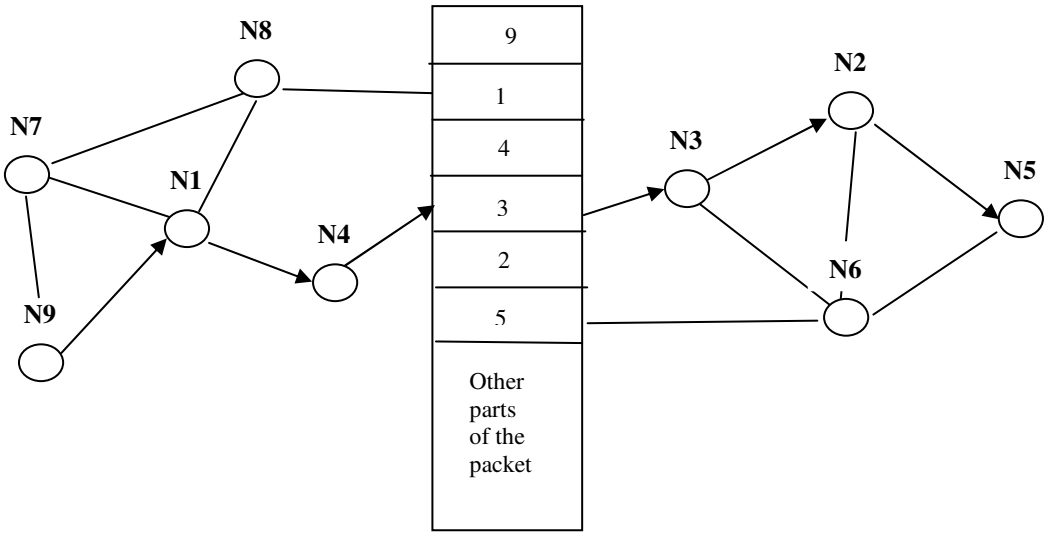


Figure 3.2 A packet being source routed from node 9 to node 5

Unlike other protocol (such as DSDV) DSR does not require the periodic transmission of router advertisements or link status packets, reducing the overhead of DSR. In addition, DSR has been designed to compute correct routes in the presence of unidirectional links.

3.5.1.Protocol Overview

Source routing is a routing technique in which the sender of a packet determines the complete sequence of nodes through which to forward the packet. The sender explicitly lists this path in the packet's header, identifying each forwarding hop by the address of the next node to which to transmit the packet on its way to the destination node.

DSR is broken down into two functional components: route discovery and route maintenance. Routing has already been described and is relatively trivial. Route discovery is the mechanism by which a node wishing to send a packet to a destination obtains a path to the destination. Route maintenance is the mechanism by which a node detects a break in its source route and obtains a corrected route.

3.5.2.Route Discovery

To perform route discovery, the source node broadcasts a route request packet with a recorded source route listing only itself. Each node that hears the route request forwards the

request (if appropriate), adding its own address to the recorded source route in the packet. The route request packet propagates hop-by-hop outward from the source node until either the destination node is found or until another node is found that can supply a route to the target.

Nodes forward route requests if they are not the destination node and they are not already listed as a hop in the route. In addition, each node maintains a cache of recently received route requests and does not propagate any copies of a route request packet after the first.

All source routes learned by a node are kept (memory permitting) in a route cache, which is used to further reduce the cost of route discovery.

A node may learn of routes from virtually any packet the node forwards or overhears. When a node wishes to send a packet, it examines its own route cache and performs route discovery only if no suitable source route is found.

Further, when a node receives a route request for which it has a route in its cache; it does not propagate the route request, but instead returns a route reply to the source node. The route reply contains the full concatenation of the recorded route from the source, and the cached route leading to the destination.

Naturally, if a route request packet reaches the destination node, the destination node returns a route reply packet to the source node with the full source to destination path listed.

3.5.3.Route Maintenance

Conventional routing protocols integrate route discovery with route maintenance by continuously sending periodic routing updates. If the status of a link or node changes, the periodic updates will eventually reflect the change to all other nodes, presumably resulting in the computation of new routes. However, using route discovery, there are no periodic messages of any kind from any of the mobile nodes. Instead, while a route is in use, the route maintenance procedure monitors the operation of the route and informs the sender of any routing errors.

If a node along the path of a packet detects an error, the node returns a route error packet to the sender. The route error packet contains the addresses of the nodes at both ends of the hop in error. When a route error packet is received or overheard, the hop in error is removed from any route caches and all routes, which contain this hop, must be truncated at that point.

There are many methods of returning a route error packet to the sender. The easiest of these, which is only applicable in networks, which only use bidirectional links, is to simply reverse the route contained in the packet from the original node. If unidirectional links are used in the network, the DSR protocol presents several alternative methods of returning route error packets to the sender.

Route maintenance can also be performed using end -to-end acknowledgments rather than the hop-by- hop acknowledgments described above. As long as some route exists by which the two end nodes can communicate, route maintenance is possible. In this case, existing transport or application level replies or acknowledgments from the original destination, or explicitly requested network level acknowledgments, may be used to indicate the status of the node's route to the other node.

In general, the two sources of bandwidth overhead in DSR are route discovery and route maintenance. These occur when new routes need to be discovered or when the network topology changes [1]. However, employing intelligent caching techniques in each node, at the expense of memory and CPU resources, can reduce this overhead. The remaining source of bandwidth overhead is the required source route header included in every packet. This overhead cannot be reduced by techniques outlined in the DSR literature.

3.6. Ad-Hoc On-Demand Distance Vector (AODV)

3.6.1.Protocol overview

AODV stands for Ad-Hoc On-Demand Distance Vector and is, as the name already says, a reactive protocol, even though it still uses characteristics of a proactive protocol.

AODV takes the interesting parts of DSR and DSDV, in the sense that it uses the concept of route discovery and route maintenance of DSR and the concept of sequence numbers and sending of periodic hello messages from DSDV.

Routes in AODV are discovered, established, and maintained only when and as long as needed. To ensure loop freedom sequence numbers, which are created and updated by each node itself, are used. These allow also the nodes to select the most recent route to a given destination node.

AODV takes advantage of route tables. In these, it stores routing information as destination and next hop addresses as well as the sequence number of a destination. Next to that, a node also keeps a list of the precursor nodes, which route through it, to make route maintenance easier after link breakage. To prevent storing information and maintenance of routes that are not used anymore each route table entry has a lifetime. If during this time the route has not been used, the entry is discarded.

3.6.2.Route Discovery

When a node wants to communicate with another node it first checks its own routing table if an entry for this destination node exists. If this is not the case, the source node has to initialize a route discovery. This is done by creating a RREQ message, including the hop count to destination, the IP address of the source and the destination, the sequence numbers of both of them, as well as the broadcast ID of the RREQ. This ID and the IP address of the source node together form a unique identifier of the RREQ. When the RREQ is created, the source node broadcasts it and sets a timer to wait for reply.

All nodes, which receive the RREQ, first check by comparing the identifier of the message with identifiers of messages already received. If it is not the first time the node sees the message, it discards silently the message. If this is not the case the node processes the RREQ by updating its routing table with the reverse route. If a node is the destination node or has already an active route to the destination in its routing table with sequence number of the destination host, which is higher than the one in the RREQ, it creates a RREP message and unicasts it to the source node. This can be done by analyzing the reverse route for the

next hop. Otherwise it increments the RREQ's hop count and then broadcasts the message to its neighbors.

When the source node receives no RREP as a response on its RREQ a new request is initialized with a higher TTL and wait value and a new ID. It retries to send a RREQ for a fixed number of times after which, when not receiving a response, it declares that the destination host is unreachable.

Fig. 3.3 shows the propagation of the RREQ through the network and the path taken by the RREP from the destination to the source node.

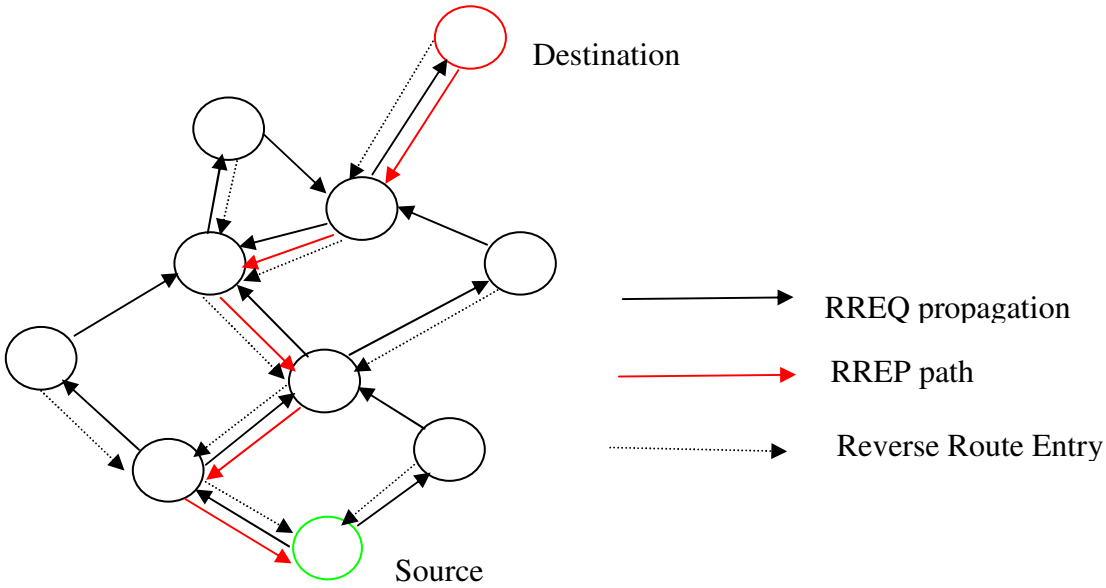


Fig 3.3 Propagation of RREQ and Route Determination

3.6.3.Route Maintenance

When a route has been established, it is being maintained by the source node as long as the route is needed. Movements of nodes affect only the routes passing through this specific node and thus do not have global effects. If the source node moves while having an active session, and loses connectivity with the next hop of the route, it can rebroadcast an RREQ. If though an intermediate station loses connectivity with its next hop it initiates an Route Error (RERR) message and broadcasts it to its precursor nodes and marks the entry of the destination in the route table as invalid, by setting the distance to infinity. The entry will only be discarded after a certain amount of time, since routing information may still be used.

When the RERR message is received by a neighbor, it also marks its route table entry for the destination as invalid and sends again RERR messages to its precursors.

In figure 3.4.I, the node N4 moves to N4' and so node N3 cannot communicate with it anymore, connectivity is lost. N3 creates a RERR message to N2, there the route is marked invalid and unicasts the message to N1. The message is unicast since we have only route passing through each node. N1 does the same thing and unicasts the message to the source node. When the RERR is received at the source node and it still needs the route to the destination it reinitiates a route discovery. Figure 3.4 II shows the new route from the source to the destination through node N5. In addition, if a node receives a data packet for a node which it does not have an active route to; it creates a RERR message
and broadcasts it as described above.

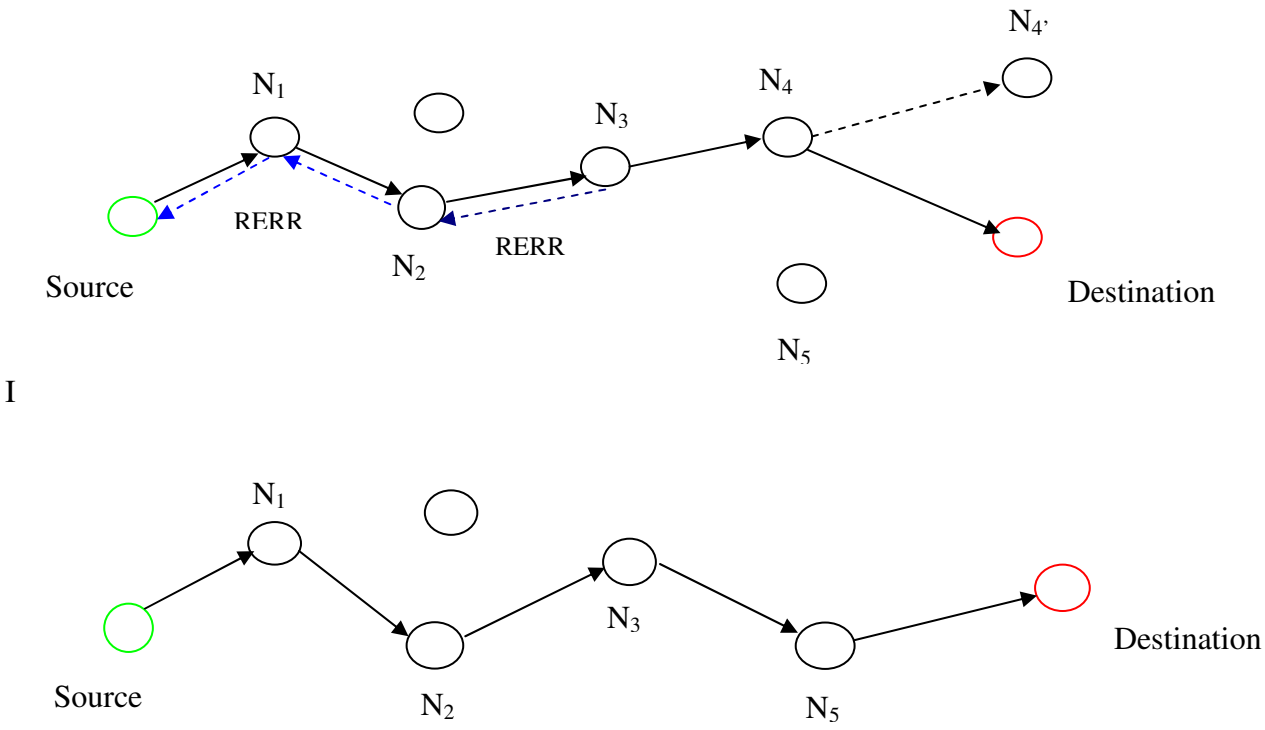


Fig 3.4 Route Maintenance

3.6.4.Hello Messages

If no broadcast has been send within, by default, one second, each node broadcasts Hello message to its neighbors in order to keep connectivity up to date. These messages contain the node's IP address and its current sequence number. So that these messages are not forwarded from the node's neighbors to third parties the Hello message has a TTL value of one.

3.7. Temporally Ordered Routing Algorithm (TORA)

Temporally Ordered Routing Algorithm (TORA) [4], [15] is a distributed routing protocol based on a "link reversal" [4] algorithm. It is designed to discover routes on demand, provide multiple routes to a destination, establish routes quickly, and minimize communication overhead by localizing the reaction to topological changes when possible. Route optimality (shortest-path routing) is considered of secondary importance, and longer routes are often used to avoid the overhead of discovering newer routes. It is also not necessary (nor desirable) to maintain routes between every source/destination pair at all times.

The actions taken by TORA can be described in terms of water flowing downhill towards a destination node through a network of tubes that model the routing state of the network. The tubes represent links between nodes in the network, the junctions of the tubes represent the nodes, and the water in the tubes represents the packets flowing towards the destination. Each node has a height with respect to the destination that is computed by the routing protocol. If a tube between two nodes becomes blocked such that water can no longer flow through it, the height of the nodes are set to a height greater than that of any neighboring nodes, such that water will now flow back out of the blocked tube and find
an alternate path to the destination.[15]

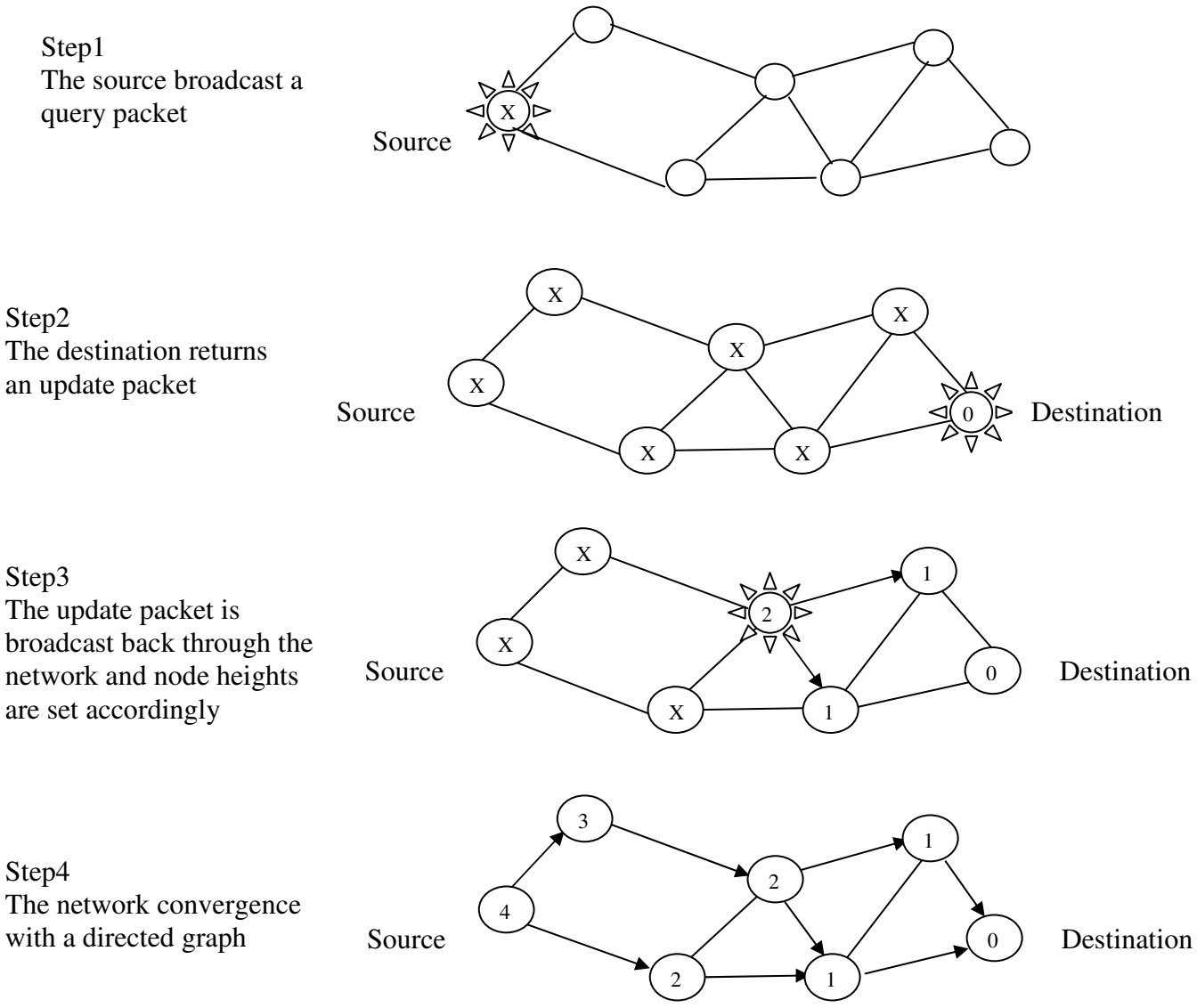


Figure 3.5 Generation of an ordered graph in TORA [4]

When a node discovers that a route to a destination is no longer valid, it adjusts its height so that it is a local maximum with respect to its neighbors and transmits an update packet. This process is illustrated in Figure 3.6.

When a node detects a network partition, where a part of the network is physically separated from the destination, the node generates a clear packet that resets the routing state and removes invalid routes from the network.

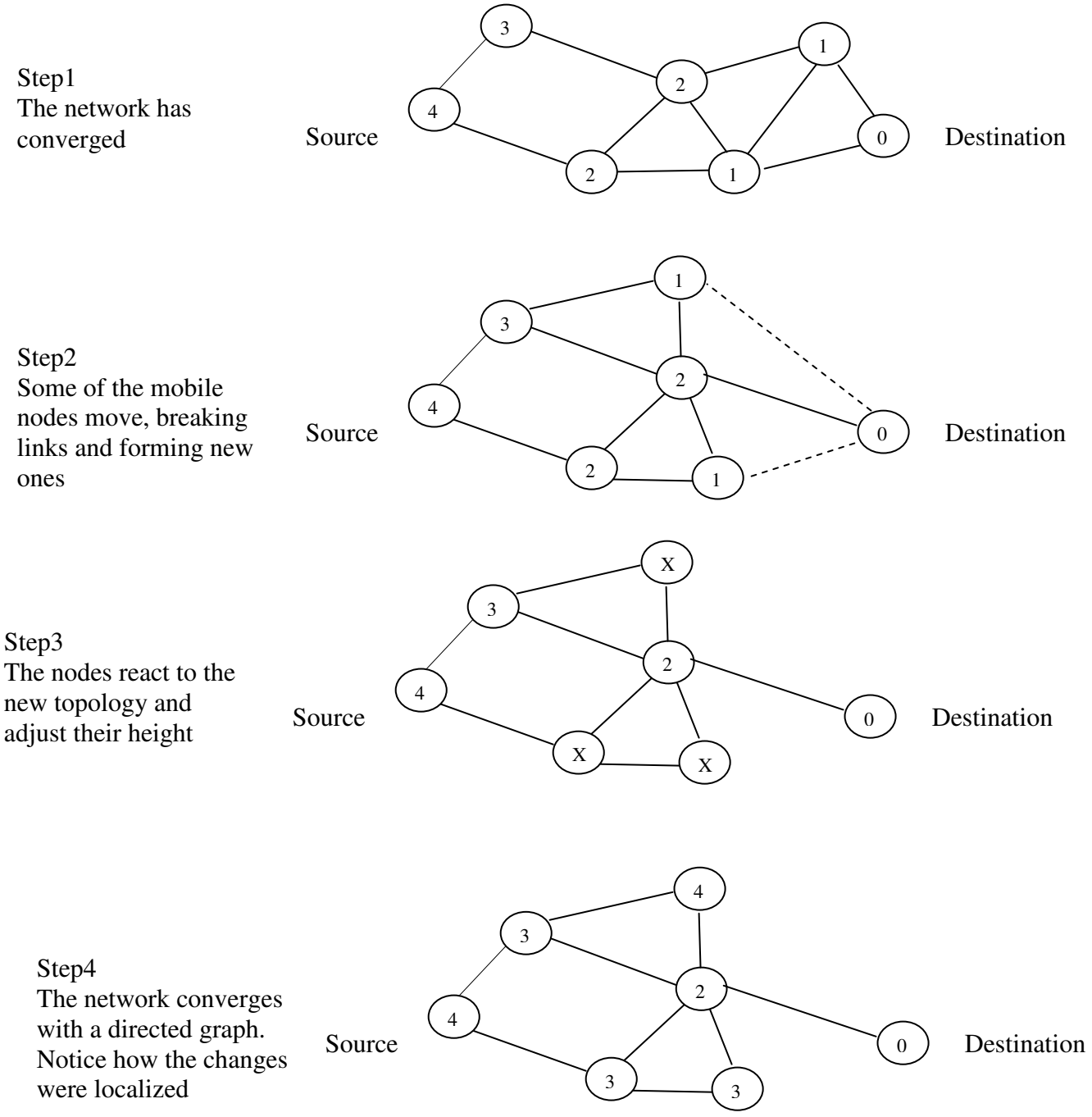


Figure3.6 TORA protocol reaction to node mobility [4]

TORA sits as a routing layer above another, network level protocol called the Internet MANET Encapsulation Protocol (IMEP) [15]. IMEP is designed to support the operation of many routing algorithms, network control protocols or other upper layer protocols intended for use in mobile ad hoc networks. The protocol incorporates mechanisms for supporting link status and neighbor connectivity sensing, control packet aggregation and

encapsulation, one-hop neighbor broadcast reliability, multipoint relaying, network layer address resolution, and provides hooks for inter -router authentication procedures.

In general, TORA is one of the largest and most complicated reactive protocols. In terms of memory requirements, each node must maintain a structure describing the node's height as well as the status of all connected links per connection supported by the network [15]. In terms of CPU and bandwidth requirements, each node must be in constant coordination with neighboring nodes in order to detect topology changes and converge. As in DSDV, routing loops can occur while the network is reacting to a change in topology [15].

3.8. Zone Routing Protocol (ZRP)

The Zone Routing Protocol (ZRP) is a hybrid reactive/proactive routing protocol which minimizes the wastage associated with pure proactive schemes by limiting the scope of the proactive procedure to a node's local neighborhood. It searches through the whole network more efficiently by querying only selected nodes in the network reactively, rather than flooding all the network nodes. Another appealing feature is that its behavior is adaptive. It can dynamically adjust itself to operational conditions based on the current configuration of network and users' behavior by sizing a single network parameter – its zone radius. Routing is flat rather than hierarchical, thus reducing overhead, allowing optimal routes to be discovered and reducing the threat of network congestion. With multiple loop-free routes to the destination identified, its reliability and performance are increased too.

The placement of ZRP in the OSI protocol stack is shown in Fig 3.7 Proactive maintenance of the routing zone topology in ZRP is performed through exchanging of update packets by a protocol called IntrAzone Routing Protocol (IARP). Such updates can be triggered by MAC-level Neighbor Discovery Protocol (NDP) which aids in informing IARP when a link to a neighbor is established or broken. Reactive routing to nodes beyond the routing zone by a query-reply mechanism is implemented by another protocol called IntErzone Routing Protocol (IERP). The following sections will further illustrate the algorithm of these protocols.

3.8.1.Routing Zone and IntrAzone Routing

A routing zone is defined as a collection of nodes whose minimum distance (in terms of hops) from the node in question is no greater than the zone radius. Note that each node will maintain its own routing zone which may overlap with neighboring nodes' (i.e., those nodes that are in direct communication with the node) routing zones too.

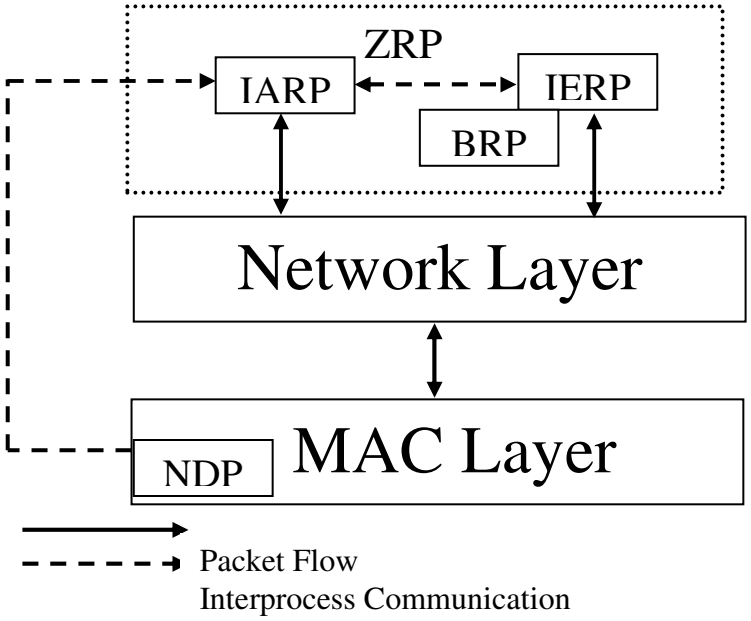


Fig. 3.7 ZRP Architecture

The concept of the routing zone is illustrated in Fig 3.8 The routing zone of central node, node S, has a zone radius of 2 hops. Nodes A to K are within the zone and therefore members of S's routing zone. However, those nodes with hops greater than 2 away from S such as node L in the figure are outside of S's routing zone. Peripheral nodes (nodes G-K) are nodes whose minimum distance to the node in question (node S) is exactly equal to the zone radius while interior nodes (nodes A-F) those remaining members which are not.

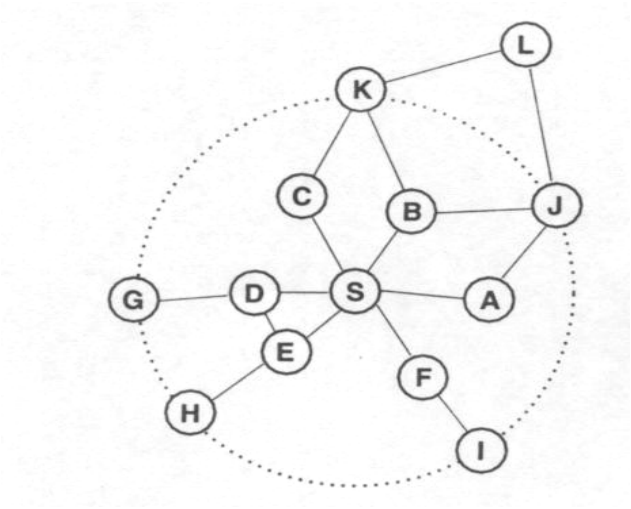


Fig. 3.8 Routing Zone of Radius 2 Hops

In ZRP, the number of routing zone nodes can be regulated through adjustments in each node's transmitter power which also determines the set of neighbor nodes, subject to local propagation conditions and receiver sensitivity. It is therefore important that a node to be connected to a sufficient number of neighbors to ensure adequate network reachability. However as the transmitters' coverage area grows larger, the membership of the routing zones will increase too, resulting in an excessive amount of update traffic and increased contention (locally) for the wireless channel.

The construction of a routing zone requires the node to know who its neighbors are. Identification can be provided by the media access control (MAC) protocols by implementing neighbor discovery through a MAC-level Neighbor Discovery Protocol (NDP). It operates by periodically broadcast "hello" beacons whereby the quality of the reception of a beacon will be used to indicate the status of the connection to the neighbor. Each node only maintains routing information to its members within its routing zone. Since the updates are propagated locally, the amount of update traffic required to maintain the zone does not depend on the total number of network nodes.

The routing zones in ZRP are maintained through a proactive component called the IntrAzone Routing Protocol (IARP). Since the protocol is mainly a framework, almost any proactive routing scheme can be modified to implement the IARP. [6].

When there are changes in the neighbor connectivity, the broadcast of IARP route update packets is triggered to reflect the new status. A node that receives such a packet will record the updated routing information and determine any changes to the shortest path to the destination. If there is, the node will then broadcast an IARP route update to inform about this change. However, the IARP differs from regular proactive schemes in that instead of

propagating updates throughout the entire network, it uses the route hop count to prevent updates from propagating beyond the zone radius, thus reducing update traffic.

3.8.2.IntErzone Routing

Whereas the IARP maintains routes to nodes within their routing zones, the IntErzone Routing Protocol (IERP) is responsible for reactively discovering routes to destinations located beyond a node's routing zone. Instead of flooding the entire network, the IERP uses a process called bordercasting. By exploiting the structure of the routing zone, this is achieved by a packet delivery service that allows a node to send a message to its peripheral nodes through a component provided, called the Bordercast Resolution Protocol (BRP).

Bordercasting can be implemented through network layer unicasting or multicasting of messages to the peripheral nodes. However, this approach prevents non-peripheral nodes from accessing the messages as they are relayed to the edge of the routing zone. Such access is central to the control of the route query process. As such, a more suitable implementation is to send messages indirectly to peripheral nodes by forwarding between BRP layers of adjacent nodes through IP unicast or IP broadcast.

The IERP route discovery works as follows. The source node checks if the destination lies within its routing zone. If the destination does not, the source will generate and bordercast a route request packet, which is uniquely, identified by a combination of the source node's ID and request number, to its peripheral nodes. Upon receipt of a route request packet, the peripheral nodes will check whether the destination is within their zone. If so, a route reply will be sent back to the source after adding its ID to the query, indicating the accumulated route from the source to the destination. If the destination does not appear in the routing zone, the node bordercasts to its peripheral nodes, which will execute the same procedure. [6]

An example of this route discovery procedure is illustrated in Fig 3.9 The source node S wants to send a packet to destination D. S first checks if D is within its routing zone. If so,

S already knows the route. Otherwise, S bordercasts a route request to all its peripheral nodes (nodes C, G and H). Now, in turn, each of these nodes determines that D is not in their routing zone and forwards the query to its peripheral nodes. In particular, H sends the query to B which recognizes D in its routing zone and responds to the request, indicating the forwarding path S-H-B-D.

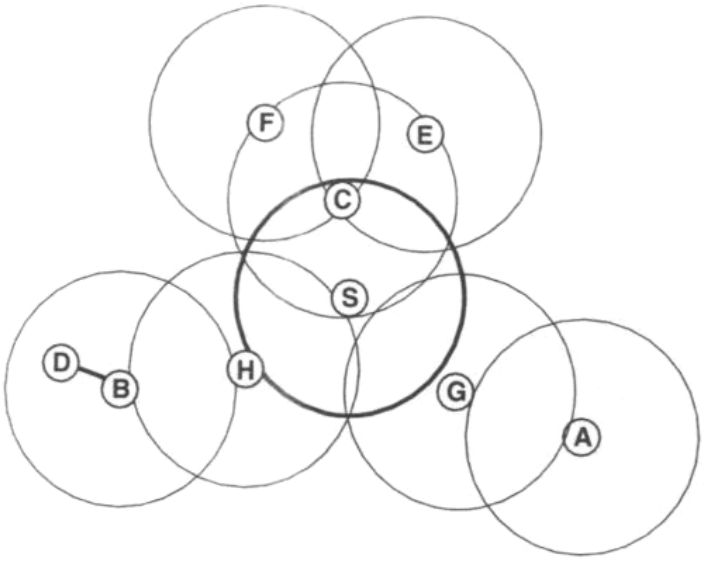


Fig.3.9 an Example of IERP Operation

One good feature of this distributed route discovery process is that a single route query can return multiple route replies which can be used to determine the best route based on relative quality (hop count or any other path metric). The intuition behind bordercasting is that querying can be performed more efficiently than flooding by directly route requests to target peripheral nodes. However, problems may arise due the fact that the routing zones are heavily overlapped. A node can be a member of many routing zones and it is likely that the same query may be forwarded by the same node multiple times, resulting in more control traffic than flooding incurs.

3.8.3.Query Control Mechanism

A node's response to a route query contains information about that node's entire routing zone. From this perspective, excess route request traffic can be regarded as a result of overlapping queried routing zones. Thus certain query control mechanisms have to be

implemented to reduce this amount of query traffic by steering the requests outward from the source's routing zone and away from each other. In this section, we will look at some of the mechanism implemented in ZRP to address this problem.

3.8.4.Loop-back Termination (LT)

Loop-back termination is an ideal mechanism to handle loop backs since it is relatively easy to identify a packet that returns to a routing zone that it previously queried, by simply examining the accumulated route in the received packet. If any hop lies within its routing zone, there is a look-back and the packet can be discarded. An illustration of this scheme is shown in Fig 3.10 Node S bordercasts a route query to A, which in turn bordercast to B. B will then bordercast to C. However, C will terminate the packet because node S, which appears in the accumulated path, also lies within C's routing zone. [17]

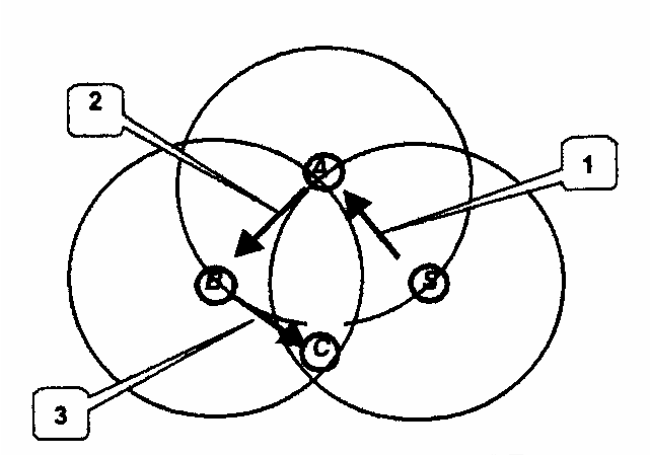


Fig. 3.10 Loop-back Termination (LT)

3.8.5.Query Detection (QD1/QD2)

When bordercasting is issued, only the node that bordercasts a route request is aware that its zone is covered by the query. When the peripheral nodes continue to bordercast to their peripheral nodes, the query may be relayed through the same nodes again. Unlike the case of loop-back, the ability to terminate in such situations depends on the ability of the nodes to detect that a routing zone that they belong to has been queried. BRP provides two query detection methods, QD1 and QD2, to notify the remaining nodes through some form of eavesdropping without incurring additional control traffic. The first level of query detection (QD1) allows the intermediate nodes (which relay queries to the edge of the routing zone) to detect these queries. In single channel networks, it is possible for queries to be detected by any node within the range of a query transmitting node. This extended query detection (QD2) can be implemented by using IP and MAC broadcasts to relay route requests.

Fig 3.11illustrates both levels of query detection. Node S bordercasts to two peri-pheral nodes, B and D. The intermediate nodes A and C can detect the passing packets using QD1. If Qd2 is implemented, node E will be able to receive A's transmission and record the query as well. A query detection table can be maintained to cache the detected queries. Foe each entry, the cache contains the address of the source node and the request ID. The unique address-ID pair can then be used to identify all queries in the network.[16], [17]

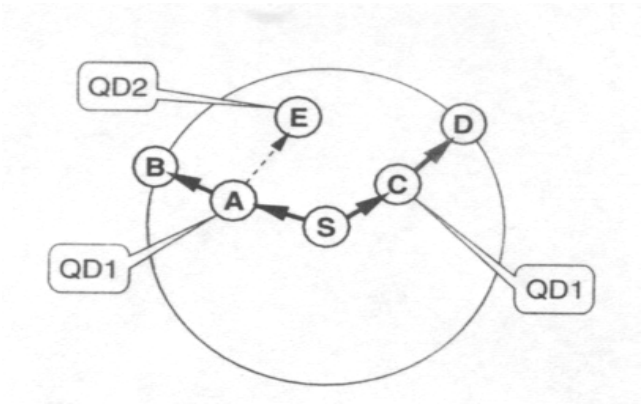


Fig. 3.11 Advanced Query Detection (QD1/QD2)

3.8.6.Early Termination (ET)

A node can prevent a route request from entering already covered regions through a scheme called early termination. Early termination uses the information from query detection to identify which local nodes have already bordercasted the query. A node can prune a peripheral node if it has already relayed a query to that node. To implement ET, topology information of an extended routing zone with radius 2*p*-1 hops (where *p* refers to the radius of the basic routing zone) of the node has to be maintained. Fig 3.12 illustrates the operation of the ET mechanism. Node S bordercasts a route request with node C as one of the intended recipients. Intermediate node A passes the query to node B. However, instead of delivering to node C, node B terminates the packet because a
different thread of this request was previously detected. [17]

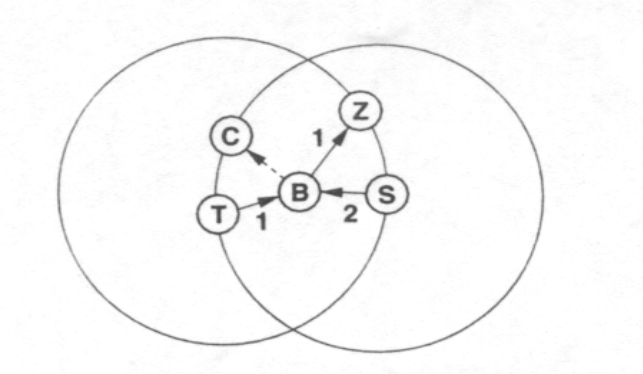


Fig. 3.12 Early Termination (ET)

3.8.7.Selective Bordercasting (SBC)

By eliminating overlap locally, some degree of control can be imposed on the direction of propagation, thereby reducing overlap further out in the network. However, rather than bordercast queries to all peripheral nodes, the same coverage can be provided by bordercasting to a chosen subset of peripheral nodes, through a mechanism called selective bordercasting (SBC).

This mechanism requires that the IARP provides network topology information for an extended zone that is twice the radius of the routing zone. A node will first determine the subset of other peripheral nodes covered by its assigned inner peripheral nodes. The node will then bordercast to this subset of assigned inner peripheral nodes which forms the minimal partitioning set of the outer peripheral nodes. Fig 3.13 illustrates this application. Node S's inner peripheral nodes are A, B and C; its outer peripheral nodes are F, G, H, X, Y and Z. From the overlapping routing zones, we can see that the 2 inner peripheral nodes of B (H and X) are also inner peripheral nodes of A and C. Node S can choose to eliminate node B from its list of bordercast recipients since A can provide coverage to F, G and H and node C can cover X, Y and Z. Overlapping queries can be prevented while still able to provide full coverage.

It should be noted that the amount of IARP traffic may increase to provide the extended zone topology. To accommodate the list of assigned peripheral nodes, the length of each

IERP route request has to increase too. Such costs have to be offset by reducing the number of query packet transmissions due to overlap prevention.[17]

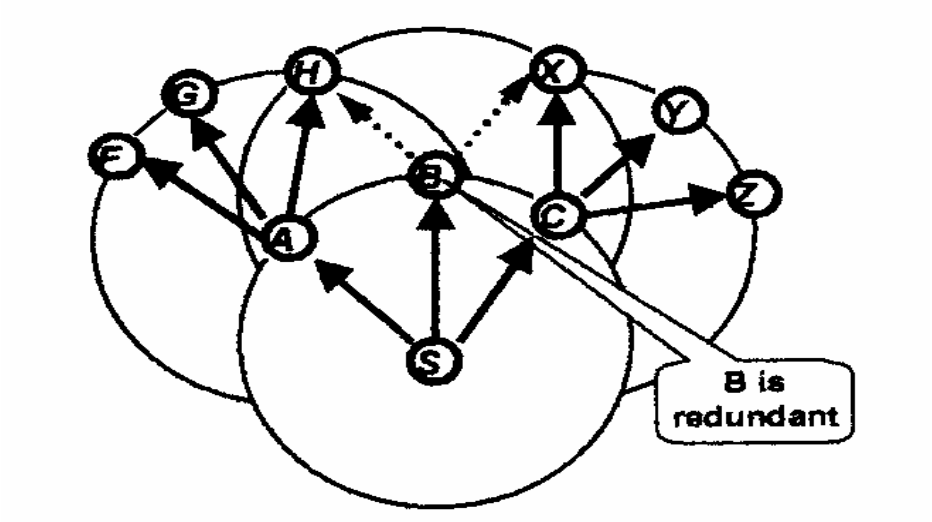


Fig. 3.13 Selective Bordercasting (SBC)

3.9. Other Routing protocols

Since Mobile Ad-hoc networks is the current research issue a number of routing algorithms proposed each time. I have study at some other routing protocols in order I can make a theoretical comparison. The following section discusses those routing protocols, which are not included in my simulation comparison.

3.9.1.Global State Routing protocol (GSR)

Global State Routing (GSR) [7] is similar to DSDV described in section 3.3. It takes the idea of link state routing but improves it by avoiding flooding of routing messages.

In this algorithm, each node maintains a Neighbor list, a Topology table, a Next Hop table and a Distance table. Neighbor list of a node contains the list of its neighbors (here all nodes that can be heard by a node are assumed to be its neighbors.). For each destination node, the Topology table contains the link state information as reported by the destination and the timestamp of the information. For each destination, the Next Hop table contains the

next hop to which the packets for this destination must be forwarded. The Distance table contains the shortest distance to each destination node.

The routing messages are generated on a link change as in link state protocols. On receiving a routing message, the node updates its Topology table if the sequence number of the message is newer than the sequence number stored in the table. After this, the node reconstructs its routing table and broadcasts the information to its neighbors.

3.9.2.Fisheye State Routing Protocol (FSR)

Fisheye routing is an improvement to GSR [9]. Amount of update messages in GSR wastes considerable amount of network bandwidth. In FSR the update message does not contain information about all nodes. Instead, it includes information about closer nodes, more frequently than it does about farther nodes, reducing the update message size. Each node gets accurate information about its neighbors.

The routing messages are generated on a link change as in link state protocols. On receiving a routing message, the node updates its Topology table if the sequence number of the message is newer than the sequence number stored in the table. After this, the node reconstructs its routing table and broadcasts the information to its neighbors.

The following figure shows the operation of this algorithm. Accuracy of information sent from the central node is decreased as going away from it.

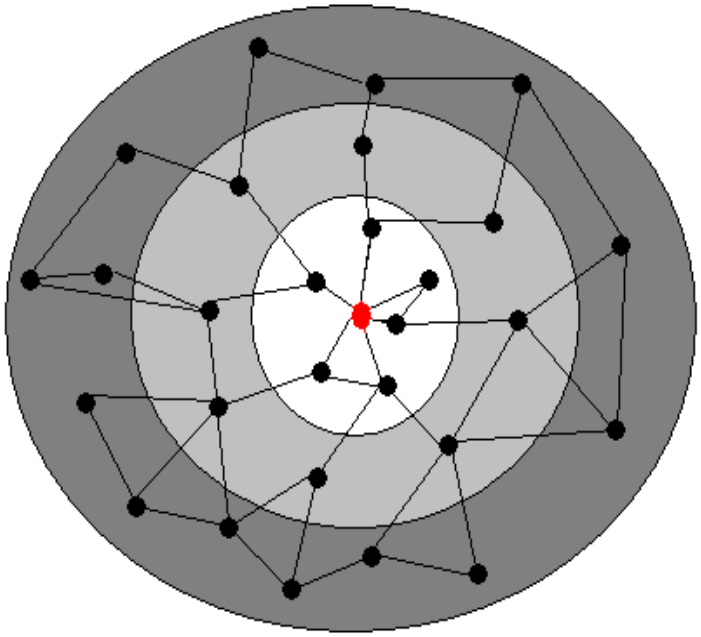


Fig 3.14 Accuracy of information in FSR [9]

3.9.3.Cluster Based Routing protocol (CBRP)

In Cluster Based Routing protocol (CBRP) [8], the nodes are divided into clusters. A cluster is formed in the following way: When a node comes up, it enters the "undecided" state, starts a timer and broadcasts a Hello message. When a cluster-head gets this hello message it responds with a triggered hello message immediately. When the undecided node gets this message it sets its state to "member". If the undecided node times out, then it makes itself the cluster-head if it has bi-directional link [18] to some neighbor otherwise it remains in undecided state and repeats the procedure again. Cluster heads are changed as infrequently as possible.

Each node maintains a neighbor table. For each neighbor, the neighbor table of a node contains the status of the link (uni- or bi-directional) and the state of the neighbor (cluster-head or member). A cluster-head keeps information about the members of its cluster and also maintains a cluster adjacency table that contains information about the neighboring clusters. For each neighbor cluster, the table has entry that contains the gateway through which the cluster can be reached and the cluster-head of the cluster.

When a source has to send data to destination, it floods route request packets (but only to the neighboring cluster-heads). On receiving the request a cluster-head checks to see if the

destination is in its cluster. If yes, then it sends the request directly to the destination else it sends it to all its adjacent cluster-heads. The cluster-heads address is recorded in the packet so a cluster-head discards a request packet that it has already seen. When the destination receives the request packet, it replies back with the route that had been recorded in the request packet. If the source does not receive a reply within a time period, it backs off exponentially before trying to send route request again.

In CBRP, routing is done using source routing [17]. It also uses route shortening that is on receiving a source route packet, the node tries to find the farthest node in the route that is its neighbor (this could have happened due to a topology change) and sends the packet to that node thus reducing the route. While forwarding the packet if a node detects a broken link it sends back an error message to the source and then uses local repair mechanism. In local repair mechanism, when a node finds the next hop is unreachable, it checks to see if the next hop can be reached through any of its neighbor or if hop after next hop can be reached through any other neighbor. If any of the two works, the packet can be sent out over the repaired path.

3.9.4.Cluster head Gateway Switch Routing Protocol (CGSR)

The mobile nodes are aggregated into clusters and a cluster-head is elected. All nodes that are in the communication range of the cluster-head belong to its cluster. A gateway node is a node that is in the communication range of two or more cluster-heads. In a dynamic network cluster head scheme can cause performance degradation due to frequent cluster-head elections, so CGSR uses a Least Cluster Change (LCC) algorithm. In LCC, cluster-head change occurs only if a change in network causes two cluster-heads to come into one cluster or one of the nodes moves out of the range of all the cluster-heads.

The general algorithm works in the following manner. The source of the packet transmits the packet to its cluster-head. From this cluster-head, the packet is sent to the gateway node that connects this cluster-head and the next cluster-head along the route to the destination. The gateway sends it to that cluster-head and so on till the destination cluster-head is

reached in this way. The destination cluster-head then transmits the packet to the destination. Figure 3.15 shows an example of CGSR routing scheme.

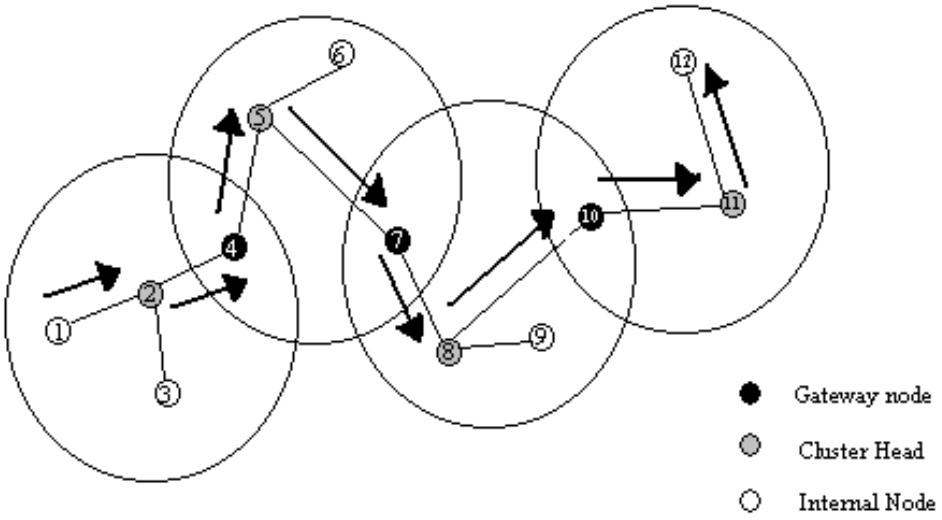


Fig 3.15 Example of CGSR routing from node 1 to node 12

Each node maintains a cluster member table that has mapping from each node to its respective cluster-head. Each node broadcasts its cluster member table periodically and updates its table after receiving other node broadcasts using the DSDV algorithm. In addition, each node also maintains a routing table that determines the next hop to reach the destination cluster.

On receiving a packet, a node finds the nearest cluster-head along the route to the destination according to the cluster member table and the routing table. Then it consults its routing table to find the next hop in order to reach the cluster-head selected in step one and transmits the packet to that node.

3.10. Theoretical Comparison of the above Routing protocols

This section summarizes the properties of the above discussed ad-hoc routing protocols.

The following tables show theoretical comparison of table-driven or proactive routing protocols and on-demand routing protocols.

3.10.1. Comparison of Table-Driven Routing Protocols

Metrics	DSDV	OLSR	GSR	FSR	CGSR
Data forwarding type	Hop-to-Hop routing	Hop-to-Hop routing	Hop-to-Hop routing	Hop-to-Hop routing	Hop-to-Hop routing
Routing metric	Shortest path	Shortest path	Shortest path	Shortest path	Shortest path (via the critical node)
Relies on Critical nodes	No	No	No	No	Yes
Loop-free	Yes	Yes	Yes	Yes	Yes
Updates transmitted to	Neighbors	To nodes selected as MPR	Neighbors	Local neighbors	Neighbors and cluster head
Frequency of update transmissions	Periodical and as needed	As needed And periodically	Periodically	Varying over scopes	Periodically
Required link status	Yes	Yes	Yes	?	?
Support Asymmetric links	No	Yes	No	No	No
Support sleeping nodes	No	No	No	No	No
Support Security	No	No	No	No	No
Use of sequence number	Yes	Yes	Yes	Yes	Yes
Scalable	No	Partially	No	Partially	Yes
Flat/hierarchal	Flat	Flat	Flat	Flat	Hierarchal

Table3.1 comparison of proactive protocols

3.10.2. Comparison of On-Demand Routing Protocols

Metric	DSR	AODV	TORA	CBRP
Data forwarding type	Source routing	Hop-to-hop routing	Hop-to-hop routing	Source routing
Routing metric	Shortest Path	Shortest Path	Shortest Path	Shortest Path
Relies on Critical nodes	No	No	No	Yes
Loop free	Yes	Yes	Yes	Yes
Route maintenance methodology	Erase route, notify source	Erase route, notify source	Link reversal Route repair	Local repair Notify source
Route maintenance	Route cache	Routing table	Routing table	Routing table
Requires link status sensing	No	Yes	Yes	?
Supports sleeping nodes	No	No	No	No
Support security	No	No	No	No
Flat/hierarchal	Flat	Flat	Flat	Hierarchal
Scalable	No	No	Partial	Yes
Support asymmetric links	Yes	No	No	Yes

Table3.2 comparison of reactive protocols

4. The simulation software

The simulator software that this research work used to simulate the various Ad-hoc routing protocols is the Network Simulator version 2(NS-2). To simulate the mobile wireless radio environment, this research has used a mobility extension to NS that is developed by the CMU Monarch project at the Carnegie Mellon University.

4.1. Network Simulator Version 2 (NS-2)

NS-2 (version 2) is an object-oriented discrete-event simulator for doing research on networks. NS provides substantial support for simulation of TCP, UDP, CBR agents routing and multicast protocols over wired and wireless networks. [19] The simulator is a result of an on-going effort of research and development. Even though there is a considerable confidence in NS-2, it is not a polished and finished product yet and bugs are being discovered and corrected continuously.

NS-2 is written in C++ with an OTCL interpreter as command and configuration interface. The C++ part which is fast to run but slower to change is used for detailed protocols implementation. The OTCL part, on the other hand, which runs much slower but can be changed very quickly, is used for simulation configuration. The simplified view of NS-2 is shown in fig 4.1.

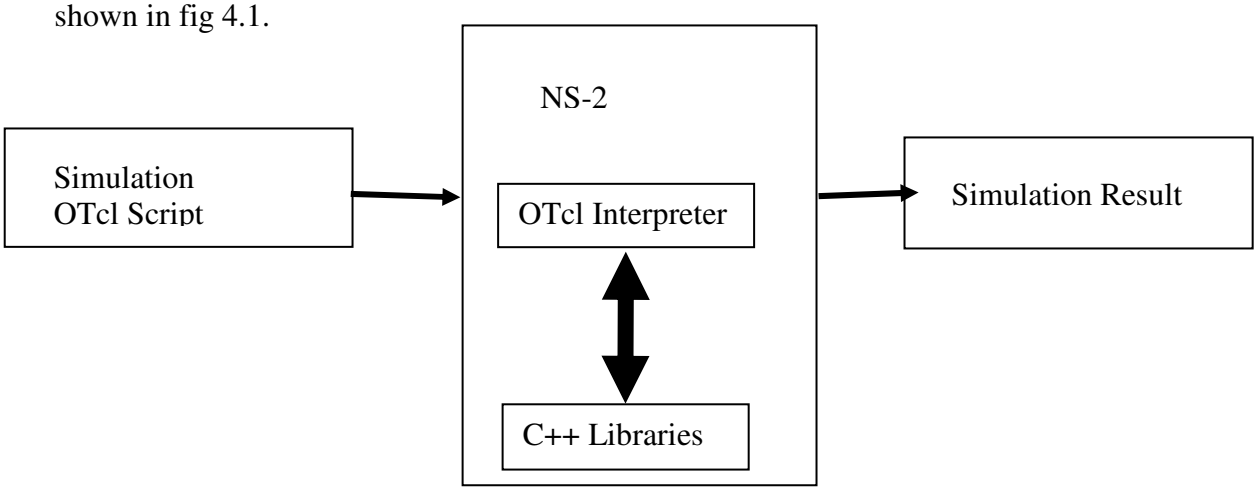


Fig 4.1 simplified user's view of NS-2

One of the advantages of split-language programming approach is that, it allows for fast generation of large scenarios. To simply use the simulator, it is sufficient to know OTCL. On the other hand, one disadvantage is that modifying and extending the simulator requires programming and debugging in both languages simultaneously.

4.2. Wireless networking in NS-2

4.2.1. Mobile Node

Wireless networking in NS-2 is defined by having a mobile node. A `Mobile Node` thus is the basic `Node` object with added functionalities of a wireless networks. Mobile nodes have the following functionalities:

- Ability to move within a given topology
- Ability to receive and transmit signals to and from a wireless channel etc.

A major difference between a node of wired network and mobile node is that a Mobile Node is not connected by means of Links to other nodes or mobile nodes. Moreover, routing in mobile networks especially in Ad-hoc networks is distributed and there is no centralized entity (as router in wired network). Therefore a mobile node acts as a router and as a node at the same time.

4.2.2. Node movement

The mobile node is designed to move in a three dimensional topology. However the third dimension (Z) is not used. That is, the mobile node is assumed to move always on a flat terrain with Z always equal to 0. Thus, the mobile node has X, Y, Z(=0) co-ordinates that is continually adjusted as the node moves.

There are two mechanisms to induce the movement in mobile nodes.

In the first method, starting position of the node and its future destinations may be set explicitly. These directives are normally included in a separate movement scenario file. The starting position and future destinations for a mobile node may be set by using the following APIs:

```
$node set X_ \<x1\>
$node set Y_ \<y1\>
$node set Z_ \<z1\>
$ns at $time $node setdest \<x2\> \<y2\> \<speed\>
```

At $time sec, the node would start moving from its initial position of (x1,y1) towards a destination (x2,y2) at the defined speed.

In this method the node-movement-updates are triggered whenever the position of the node at a given time is required to be known. This may be triggered by a query from a neighboring node seeking to know the distance between them, or the setdest directive described above that changes the direction and speed of the node.

An example of a movement scenario file using the above APIs, can be found in tcl/mobility/scene/scen-500x500-50-50-20-0. Here, 500x500 defines the length and width of the topology with 50 nodes moving at a maximum speed of 20m/s with an average pause time of 50s. These node movement files may be generated using CMU's scenario generator which is found under *indep-utils/cmu-scen-gen/setdest.*

The second method employs random movement of the node. The primitive to be used is:

```
$mobilenode start
```

which starts the mobile node with a random position and have routined updates to change the direction and speed of the node. The destination and speed values are generated in a random fashion. The mobile node movement is implemented in C++.

Irrespective of the methods used to generate the node movement, the topography for mobile nodes needs to be defined. It should be defined before creating the mobile nodes. Normally, flat topology is created by specifying the length and width of the topography using the following primitive:

```
set topo        [new Topography]
$topo load_flatgrid $opt(x) $opt(y)
```

where opt(x) and opt(y) are the boundaries used in simulation.
The movement of the node is determined by setting an area and mobility scenarios.

4.2.3.Network Components in a mobile node

The network stack for a mobile node consists of a link layer (LL), an ARP module connected to LL, an interface priority queue (IFq), a MAC layer (MAC), a network interface (netIF), all connected to the channel. These network components are created and plumbed together in OTcl.

Each component is discussed briefly as follows:

Link Layer

The `LL` object is responsible for simulating the data link protocols like in wired networks. The only difference being the link layer for mobile node has an ARP module connected to it which resolves all IP to hardware (Mac) address conversions. Normally for all outgoing (into the channel) packets, the packets are handed down to the `LL` by the Routing Agent. The `LL` hands down packets to the interface queue. For all incoming packets (out of the channel), the mac layer hands up packets to the `LL` which is then handed off at the `node_entry_` point. [19]

ARP

The Address Resolution Protocol module receives queries from Link layer. If ARP has the hardware address for destination, it writes it into the Mac header of the packet. Otherwise it broadcasts an ARP query, and caches the packet temporarily. For each unknown destination hardware address, there is a buffer for a single packet. In case, additional packets to the same destination are sent to ARP, the earlier buffered packet is dropped. Once the hardware address of a packet's next hop is known, the packet is inserted into the interface queue. [19]

Interface Queue

The PriQueue../ns-2/priqueue.h [19] is implemented as a priority queue which gives priority to routing protocol packets, inserting them at the head of the queue. It supports running a filter over all packets in the queue and removes those with a specified destination address. [19]

Mac Layer

The IEEE 802.11 distributed coordination function (DCF) Mac protocol has been implemented by CMU. It uses a RTS/CTS/DATA/ACK pattern for all unicast packets and simply sends out DATA for all broadcast packets. The implementation uses both physical and virtual carrier sense. [19]

Tap Agents

`Agents` that subclass themselves as Tap../ns-2/mac.h [19] defined in mac.h can register themselves with the mac object using method installTap(). If the particular Mac protocol permits it, the tap will promiscuously be given to all packets received by the MAC layer, before address filtering is done. [19]

Network Interfaces

The Network Interface layer serves as a hardware interface which is used by mobile node to access the channel. The wireless shared media interface is implemented as Phy/WirelessPhy../ns-2/wireless-phy.h [19]. This interface subject to collisions and the radio propagation model receives packets transmitted by other node interfaces to the channel. The interface, stamps each transmitted packet with the meta-data related to the transmitting interface like the transmission power, wavelength etc. This meta-data in packet header is used by the propagation model in receiving network interface to determine if the packet has minimum power to be received and/or captured and/or detected (carrier sense) by the receiving node. The model approximates the DSSS radio interface (Lucent WaveLan direct-sequence spread-spectrum). [19]

Radio Propagation Model

It uses Friss-space attenuation () at near distances and an approximation to Two ray Ground () at far distances. The approximation assumes specular reflection off a flat ground plane. See tworayground.{cc,h}in [19].

Antenna

An omni-directional antenna having unity gain is used by mobilenodes. See antenna.{cc,h} for implementation details in[19].

4.2.4.Routing Protocols or Agents

There are four different types of Ad-hoc routing agents defined by NS-2 currently. These are:

- DSDV (Destination Sequenced Distance Vector) discussed in chapter 3
- DSR (Dynamic Source Routing) discussed in chapter 3
- AODV(Ad-hoc On-Demand Distance Vector) discussed in chapter 3 and
- TORA (Temporal Ordered Routing Algorithm) discussed in chapter 3

4.3. NS-2 Trace Support

The NS-2 trace files are used for post processing the simulation that is carried on. The following are the supported formats of trace files:

- Trace files for wired networks that is wired
- Satellite
- Wireless (old and new trace)
- Wired-cum-wireless

In the simulation, this research has used the new trace format for wireless networks. And the following section describes the new trace format. [19]

```
s -t 0.267662078 -Hs 0 -Hd -1 -Ni 0 -Nx 5.00 -Ny 2.00 -Nz 0.00 -Ne
-1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 0.255 -Id -
1.255 -It message -Il 32 -If 0 -Ii 0 -Iv 32
s -t 1.511681090 -Hs 1 -Hd -1 -Ni 1 -Nx 390.00 -Ny 385.00 -Nz 0.00
-Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 1.255 -Id
-1.255 -It message -Il 32 -If 0 -Ii 1 -Iv 32
s -t 10.000000000 -Hs 0 -Hd -2 -Ni 0 -Nx 5.00 -Ny 2.00 -Nz 0.00 -Ne
-1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 0.0 -Id 1.0 -
It tcp -Il 1000 -If 2 -Ii 2 -Iv 32 -Pn tcp -Ps 0 -Pa 0 -Pf 0 -Po 0
r -t 10.000000000 -Hs 0 -Hd -2 -Ni 0 -Nx 5.00 -Ny 2.00 -Nz 0.00 -Ne
```

The new trace format as seen above can be divided into the following fields:

Event type

In the traces above, the first field (as in the older trace format) describes the type of event taking place at the node and can be one of the four types:

S	Send
R	Receive
D	Drop
F	Forward

General tag

The second field starting with "-t" may stand for time or global setting

-t	Time
-t	Global setting

Node property tags

This field denotes the node properties like node-id, the level at which tracing is being done like agent, router or MAC. The tags start with a leading "-N" and are listed as below:

-Ni	Node id
-Nx	Node's x-coordinate
-Ny	Node's y-coordinate
-Nz	Node's z-coordinate
-Ne	Node's energy level
-Nl	Trace level, such as AGT,RTR,MAC
-Nw	Reason for event

The different reasons for dropping a packet are given below:

END	DROP_END_OF_SIMULATION
COL	DROP_MAC_COLLISION
DUP	DROP_MAC_DUPLICATE
ERR	DROP_MAC_PACKET_ERROR
RET	DROP_MAC_RTERY_COUNT_EXCEDDED
STA	DROP_MAC_INVALID_STATE
BSY	DROP_MAC_BUSY
NRTE	DROP_RTR_NO_ROUTE i.e no route is available
LOOP	DROP_RTR_ROUTE_LOOP i.e there is a routing loop
TTL	DROP_RTR_TTL i.e TTL has reached zero

TOUT	DROP_RTR_QTIMEOUT i.e packet has expired
CBK	DROP_RTR_MAC_CALLBACK
IFQ	DROP_IFQ_QFULL i.e no buffer space in IFQ
ARP	DROP_IFQ_ARQ_FULL i.e dropped by ARP
OUT	DROP_OUTSIDE_SUBNET i.e dropped by base stations on receiving routing updates from nodes outside its domain

Packet information at IP level

The tags for this field start with a leading "-I" and are listed along with their explanations as following:

-Is	Source address.source port number
-Id	Dest address.dest port number
-It	Packet type
-Il	Packet size
-If	Flow id
-Ii	Unique id
-Iv	TTL value

Next hop info

This field provides next hop info and the tag starts with a leading "-H".

-Hs	Id for this node
-Hd	Id for next hop toward the destination

Packet info at MAC level

This field gives MAC layer information and starts with a leading "-M" as shown below:

-Ma	Duration
-Md	Destination Ethernet address
-Ms	Source Ethernet address
-Mt	Ethernet type

Packet info at "Application level"

The packet information at application level consists of the type of application like ARP, TCP, the type of ad-hoc routing protocol like DSDV, DSR, AODV etc being traced. This field consists of a leading "-P" and list of tags for different applications are listed as below:

-P arp	Address Resolution Protocol
-Po	ARP Request/Reply
-Pm	Source MAC address
-Ps	Source address
-Pa	Destination MAC address
-Pd	Destination address
-P dsr	DSR routing protocol (an example DSR and the following details for DSR)
-Pn	How many nodes traversed
-Pq	Routing request flag
-Pi	Route request sequence number
-Pp	Route reply flag
-Pl	Reply length
-Pe	Source of source routing to destination
-Pw	Error report flag
-Pm	Number of errors
-Pc	Report to whom
-Pb	Link error form linka->linkb
-P cbr	Constant bit rate information and the following are details of CBR
-Pi	Sequence number
-Pf	How many times his packet was forwarded

-Po	Optimal number of forwards
-P tcp	Information about TCP flow and the following are details of TCP
-Ps	Sequence number
-Pa	Acknowledgment number
-Po	Optimal number of forwards
-Pf	How many times this packet was forwarded

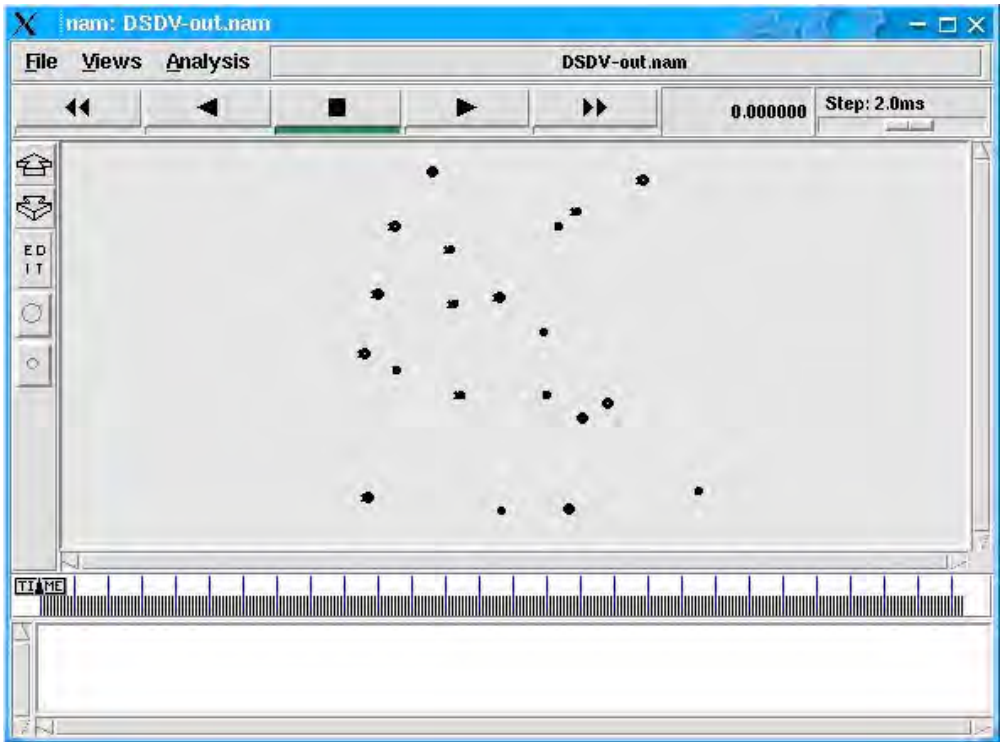
The new trace format is also under improvement to support other detailed information.

4.4. NS-2 Network Animator (NAM)

Network Animator (NAM) is an animation tool for viewing network simulation traces and real world packet traces [19]. It supports topology layout, packet level simulation and various data inspection tools.

Before starting to use NAM, a trace file has to be created. This trace file is usually generated by NS-2 as discussed above. It contains topology information for example node and links as well as packet losses. During simulation, the user can produce topology configuration, layout information and packets traces using tracing events in NS-2.

Once the trace file is generated, NAM can be used to animate it. Upon starting, NAM will read the trace file, create the topology, pop up a window, do layout if necessary and pause at time 0. Through its user interface, NAM provides control over many aspects of



animation. The following figure shows a screenshot of NAM window simulating wireless networks.

Fig 4.2 NS-2 Network Animator window

5. Implementation of Zone Routing Protocol

The NS-2 simulation software provides a useful modeling tools to simulate common network protocols. Among the many features provided by NS-2, ad-hoc routing protocols are the main ones. As discussed in the literature part of this thesis, the two main categories of MANET routing protocols can't adequately satisfy the characteristics and requirements of an ad-hoc network. A new variant of MANET routing protocol, which takes advantages from each of the proactive and reactive categories, has been proposed under the term Zone Routing Protocol (ZRP). Though NS-2 supports a number of routing protocols from both reactive and proactive categories, it doesn't support this new hybrid routing protocol – ZRP.

While evaluating the various MANET routing protocols, one major task of this thesis was to implement and analyze ZRP under NS-2. The NS-2 implementation of ZRP greatly depends on the information specified under IETF RFC document. As mentioned in the RFC, the ZRP is not so much a distinct protocol; rather it provides a framework for other protocols: IARP, BRP and IERP.

The IARP (IntrAzone Routing Protocol) pro-actively and periodically distributes route information among the members of a zone, which are defined as the nodes that are upto and including radius R hops away. This parameter R singularly defines the behavior of the overall ZRP protocol.

In this project the IARP relies on information from the NDP (Neighbor Discovery Protocol) to track neighbors and link states with neighbors. NDP uses a beacon and acknowledgment packets to obtain link states. New neighbors or expired entries in the neighbor table trigger an immediate update to the entire zone and periodic link updates keep the entire zone appraised of each other's states.

The prime directive of IARP is to provide a readily available route when a packet needs to be routed to a destination within the zone. But when a packet arrives for which no local route exists, the IERP is in charge of finding the so-called outer route, or external route.

The IERP (IntErzone Routing Protocol) sends out requests that are forwarded, accumulating the route along the way, until a forwarding node, or relay, detects that the destination is within its zone. Then a reply is created with the full route and sent back to the source.

The flood of packets from this route discovery process can overwhelm a network, which is where the BRP (The Bordercasting Resolution Protocol) steps in. BRP sends, or bordercasts, IERP requests only to peripheral nodes, also called bordercasters, which in turn forward to their bordercasters.

BRP further steers outgoing queries by using QD (Query Detection) and Early Termination. Queries destined for a previously queried bordercaster, or a previously queried routing zone, are discarded. Queries are detected as they traverse a network, and this information is used to (ET) terminate redundant trajectories.

5.1. Design

While implementing ZRP, the concepts discussed earlier are modeled as shown below in the figure. Figure 5.1 is a schematic representation of ZRP protocols showing their interaction with that of OSI layer 3 and layer 2 protocols.

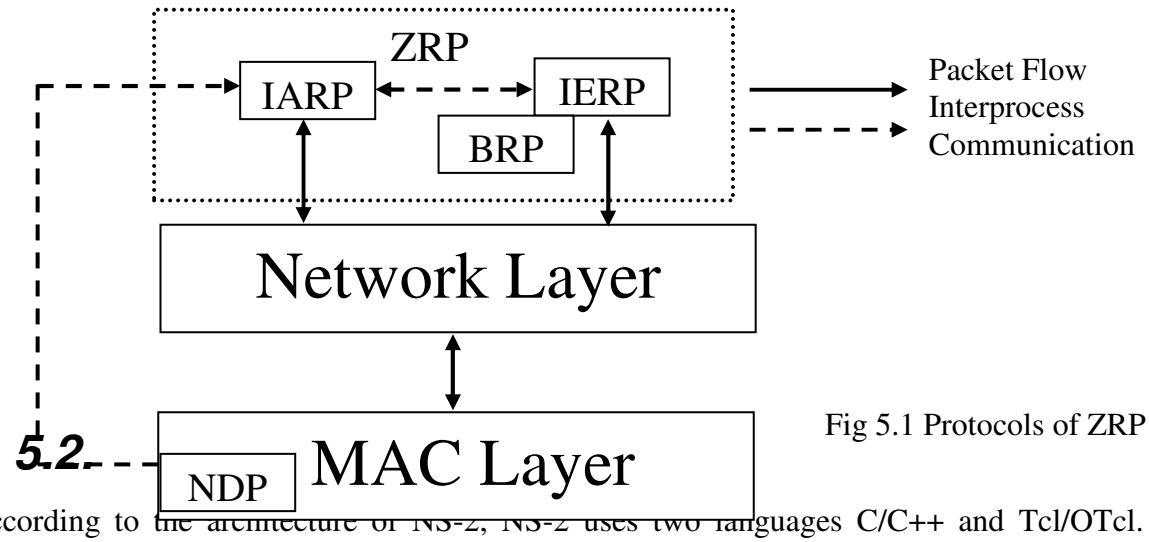


Fig 5.1 Protocols of ZRP

According to the architecture of NS-2, NS-2 uses two languages C/C++ and Tcl/OTcl. Simulation of protocols requires efficient manipulation of bytes and packet headers making the run-time speed very important where C/C++ is used. Parameter changing and

examining the scenario very quickly is best achieved by OTcl. The implementation of a new routing protocol, ZRP, for NS-2 is achieved by using a low level programming C and Gnu C compiler version 2.95.3 (is used). The platform used while developing the project is Intel PIV Desktop installed with SuSE Linux Version 8.0. The developed ZRP code can easily be installed on version 2.26 of NS-2 and of course, it can also work on latest versions of NS-2, which are to be released.

5.3. Implementation

The ZRP module was designed within the context of the mobile node architecture of the Monarch (Mobile Network Architecture) Project, a joint project between Rice University and CMU. Their software is considered as an extension of the NS-2 code-base. With this architecture of mobile node, all packets received by the node go to the routing agent. Following the same approach, the new routing agent for ZRP, `ZRPAgent` is implemented. The class `ZRPAgent`, which inherits from the main NS-2 class `Agent`, is responsible for managing all packets received by the mobile nodes.

Below is a partial[1] representation of the class declaration for ZRPAgent

```
class ZRPAgent : public Agent {
      public:
            ZRPAgent(nsaddr_t id);//constructor ZRP routing agent
      //==Timers ==
      ZRPNeighborScanTimer    NeighborScanTimer_;
      ZRPBeaconTransmitTimer  BeaconTransmitTimer_;
      ZRPPeriodicUpdateTimer  PeriodicUpdateTimer_;
      ZRPAckTimer AckTimer_;
```

[1] Only those main declarations are displayed here. This is simply to reduce the size of the report

```
        //==Methods ==
        void recv(Packet * p, Handler *); //Method for receiving
                                          //packets by ZRP Agent

        void recvZRP(Packet* p);      // Method for receiving ZRP
                                      // type packets
        // Method for Scanning IERP Cache. If any IERP requests
        // time out, delete them.
        void scan_ierp_cache();

        // Method for routing packets. Before routing first, the
        // method is first check if there is a local route. If so,
        // add route to packet and send.
        // Otherwise check if there is an outer route. If so, add
        // and send. Otherwise, originate IERP request.
        void route_pkt(Packet* p, nsaddr_t dest);

        // Method to register a neighbor with address "addr" to NDP
        // table. The neighbor is to be detected with NDP beacons.
        int  neighbor_add(nsaddr_t newneighbor);

        // Method to scan query detect table
        // and delete expired entries.
        void scan_query_detections();


        // Method to broadcast IARP packet to all neighbors
        void pkt_broadcast(Packet *p);


        // Method to create a fresh new packet and initialize it
        // with ZRPTYPE packet type, source address and its TTL
        // value
        Packet* pkt_create(ZRPTYPE zrp_type, nsaddr_t addressee,
                           int ttl);

        // Method to dynamically allocate memory for route data in
        // a packet with given “size” number of nodes in route
        // list
        void pkt_create_route_data_space(Packet *p, int size);


        // Method to free allocated memory for route data in a
        // packet. This one is to reduce memory requirement
        void pkt_free_route_data_space(Packet *p);


        // Wrapper for drop() which cmutrace.cc of NS-2 trace has.
        // A method which is responsible for dropping packet after
        // its memory space is properly freed
        void zdrop(Packet *p, const char *s);

        //==Data ==
        int myaddr_;   // variable storing an Agent’s node address
        int radius_;   // Can be set from Tcl script, see
        int beacon_period_; // Can be set from Tcl script, see
};
```

The heart of the ZRPAgent are the `recv()` and `recvZRP()` functions. The `recv()` function handles packets received from the upper and lower layers. If the packet is from the upper layer, it calls `route_pkt()`, which is considered IERP, to find out if it can route it locally through IARP by calling `add_local_route()`. If there is no local route, `route_pkt()` checks if there is a cached outer route by calling `add_outer_route()`. If so, the route is appended and sent. Otherwise, `route_pkt()` originates a request by calling the `originate_request()` method, which unicasts (boardercasts) an IERP route request to each peripheral node.

```
Void ZRPAgent::recv(Packet * p, Handler *)  {

  hdr_ip  *hdrip = HDR_IP(p);
  hdr_cmn *hdrc = HDR_CMN(p);
  hdr_zrp *hdrz = HDR_ZRP(p);
  int src = Address::instance().get_nodeaddr(hdrip->saddr());
  int dst = hdrc->next_hop();
  int jem;
  Time now = Scheduler::instance().clock(); // get the time

  rx_++; // just received a new packet

  // check if the node is asleep (simply saving energy)
  if (suspend_flag_ == TRUE) {
        zdrop(p, DROP_RTR_IARP);
      return;
  }

  if (hdrc->ptype() == PT_ZRP) {
    jem =  hdrip->ttl();
    hdrip->ttl() -= 1;
    assert(jem == (hdrip->ttl() + 1) );
    if (hdrip->ttl() < 0) {
      zdrop(p, DROP_RTR_TTL);
      return;
    }

    // it is a proper ZRP packet, let ZRP process it
    recvZRP(p);
    return;
  }

  // if other type
  if(src == myaddr_ && hdrc->num_forwards() == 0) {
    // A packet from higher layer
    hdrc->size() += IP_HDR_LEN;     //  Add the IP Header size
    //   hdrip->ttl_ = IP_DEF_TTL;
    hdrip->ttl() = radius_ ;
  } else if (src == myaddr_) { // rcving a pkt I sent, probably
                               // a routing loop
    zdrop(p, DROP_RTR_ROUTE_LOOP);
```

```
    return;
  } else {

    // forwarding a non-ZRP packet from another node
    // Check the TTL.  If it is zero, then discard.
    if(--hdrip->ttl() == 0) {
      zdrop(p, DROP_RTR_TTL);
      return;
    }

  }

  // This packet is not a ZRP packet, it needs to be routed.
  route_pkt(p, hdrip->daddr() );
}
```

Once a packet is received, the next step is to identify the type of packet and then route it through the appropriate path. This task is what the **`route_pkt(Packet* p, nsaddr_t daddr)`** method tries to do.

```
void ZRPAgent::route_pkt(Packet* p, nsaddr_t daddr) {
  hdr_cmn *hdrc = HDR_CMN(p);
  hdr_ip *hdrip = HDR_IP(p);
  hdr_zrp *hdrz = HDR_ZRP(p);
  nsaddr_t *nexthop;
  int st;
  int ierpd;
  Time now = Scheduler::instance().clock(); // get the time

  hdrz->routeindex_=0;
  hdrz->numentries_ =0;

  st = add_local_route(p,daddr);
  if (!st) {
    st = add_outer_route(p,daddr);
    nsaddr_t *rrr = hdrz->route_;
  }

  if (st) { // if a local or outer route exists, it was added,
            // start sending...
    //send to first addr on list
    //nexthop = (nsaddr_t *)p->accessdata();
    nexthop = hdrz->route_;

    // this packet will now be encapsulated into an IARP_DATA
    // packet save upper layer info in encapsulated area
    hdrz->enc_daddr_ = hdrip->daddr() ;
    hdrz->enc_dport_ = hdrip->dport() ;
    hdrz->enc_ptype_ =  hdrc->ptype();

    hdrc->ptype() = PT_ZRP;
    hdrc->next_hop() = nexthop[0];
    hdrc->addr_type_ = NS_AF_NONE;
    hdrc->size() = IP_HDR_LEN ;  //  set default packet size
```

```
    hdrip->ttl() = IERP_TTL;
    hdrip->daddr() = nexthop[0] ;
    hdrip->dport() = ROUTER_PORT;
    hdrz->zrptype_ = IARP_DATA;
    hdrz->originator_ = myaddr_;
    hdrz->pktsent_ = Scheduler::instance().clock(); // get the
                                                    // time
    hdrz->seq_ = seq_ ;
    inc_seq();
    send_next_hop(p);
    return;
  }

  // no local or cached route, call IERP
  // push onto ierp request queue
  int flag = FALSE;
  ierpd = hdrz->ierpdest_;

  for (int i=0; i<MAX_IERP_CACHE; i++) {
    if (ierpcache[i] == NULL) {
      flag = TRUE;
      st = i;
    } else if (HDR_ZRP(ierpcache[i]->pkt_)->ierpdest_ == ierpd) {
      // ierp request pending
      // drop packet
      printf("\n_%2d_ [%6.f] | A request for destination already
in "
           "ierpcache, dropping packet\n",myaddr_,now);
      zdrop(p,DROP_RTR_IERP);
      // or add buffer data structure
      return;

    }

  }
  assert(flag == TRUE); // increase MAX_IERP_CACHE otherwise
  if (flag == FALSE) {  // don't like assertion? comment out and
                        //keep this
    zdrop(p,DROP_RTR_HIYA);
    return;
  }

  // new entry for ierp request cache
  ierpcache[st] = new IERPCacheEntry;
  ierpcache[st]->qid_ = qid_;
  ierpcache[st]->expiry_ = Scheduler::instance().clock() +
    IERP_REQUEST_TIMEOUT;
  ierpcache[st]->pkt_ = p;
  originate_request(daddr, qid_ ) ;
  qid_++;

}
```

The above methods are few of the methods that handle and process a ZRP packet received from upper layer – carrying user data. But the received packet may come from the lower

layers, in this case method recvZRP() takes action instead of method recv(). This method is a large switch statement with cases for each type of ZRP packet: NDP_BEACON, NDP_BEACON_ACK, IARP_UPDATE, IARP_DATA, IERP_REQUEST and IERP_REPLY. Each case will either cache the packet for the future, drop it, or re-send it.

For NDP, recvZRP() processes beacons by sending an acknowledgement, re-using the packet object just received – node address, setting TTL to 1 because the next node should drop the ACK packet after processing it. Upon receiving an NDP_BEACON_ACK, a reachable node is obtained in the range of the zone radius, NDP (recvZRP()) checks if the neighbor is in its neighbor table. If it is not, NDP adds it and also updates the link state table and route table. The existence of a new neighbor is to be advertised immediately through IARP update.

All IARP_UPDATEs are processed the same way. First the link state table is updated. If IARP as recv() finds there was a change, it re-computes the route table. IARP forwards the update by broadcast if the TTL is not zero.

IERP_REQUEST packets will encounter nodes that will identify themselves as either relays or as a bordercaster. First the recvZRP() by using BRP s to see if the destination is in its zone. If so, an IERP_REPLY is created by appending the full route, then finally the reply is sent to the previous hop via send_prev_hop(). If the destination is not in the current zone, the node decides whether it is a relay or bordercaster by inspecting the route list. The last route entry of a request by definition will name the next bordercaster, so the current node can decide if it is a relay or bordercaster by testing if its id matches with the last entry.

If the node is a relay, the packet is sent by unicast to the next bordercaster if it is not on the covered list. If the next bordercaster is on the covered list and is part of Early Termination, then no packet is to be sent. The covered list is found by looking at the query by the unique (query id, origin) pair in the queries_ table. The relay adds the nodes internal to the last bordercaster's zone, not including its peripheral nodes, by calling query_detect1(). Finally, the next bordercaster is added to the covered list.

If the node is a bordercaster, the node first marks all the nodes in the last bordercaster's zone as covered. The node then unicasts a copy of the request to each of its peripherals if that peripheral is not on its covered list, Early Terminating those that are. After the bordercast is done, the node marks the nodes of its own zone as covered.

When recv() gets an IERP_REPLY, it is as IERP, which then relays the packet to the previous hop by send_prev_hop() or hands the packet off to IARP via ierp_notify_iarp() if the node is the originator of the original request. The outer route is cached with an expiry time in a cache to be re-used for outer-routing packets sent in the near future.

The IARP_DATA packet encapsulates the upper layer packet. The IARP_DATA packet is forwarded hop-by-hop, IERP-to-IERP, to the destination ZRPAgent, where it is unwrapped by recvZRP() (ie the original port and packet type are placed in the header) and sent upward to its upper layer.

The timer events trigger actions that will lead to a packet transmission (beacon transmission and IARP updates) or the scanning and purging of expired data entries in tables (scan IERP cache, scan query detect table and scan neighbor table). All these timers re-schedule themselves at the end of the handle() method, with the exception of the acktimer, which is only scheduled by NDP when a beacon is broadcast.

The method below shows the skeleton of recvZRP().

```
Void ZRPAgent::recvZRP(Packet* p){
      //list of variable declarations
      …
      …
      …
      Time now = Scheduler::instance().clock(); // get the time
      switch (ZRPPACKETTYPE)  {

            case NDP_BEACON: // getting a beacon, send ack
                             // re-use packet to create beacon
                             // ack, no need to drop
                  hdrip->ttl() = 1;       // Need to be dropped
                                          // after next hop

                  // Prepare packet source address
                  // determine next hop address (directly to the
                  // originator)
                  …
                  …
                  …
```

```
      pkt_send(NDP_BEACON_ACK);  // send ack

      break;

case NDP_BEACON_ACK: // getting an ack, update
                     // neighbor table and send
                     // update, and update link state
                     // table.
      // check if the ack is for the beacon I sent
      if (Am I the originator ?) {

            // add, or update if already in table
            st = neighbor_add(hdrip->saddr() );
      } else { // The ack is not ment for me
            //Drop the packet
            zdrop(p,DROP_RTR_IARP);

            return; // No need to continue
      }
      // Since the packet is ment for me, after
      // updating my route information, I have to
      // inform my neighbors

      …
      …
      …

      break;
case IARP_UPDATE:  // some body from my neighbor
                   // wants to tell me the discovery
                   // of a new neighbor received it
                  // and forward if appropriate.
      // Methods to update routing table
      …
      …
      …
      // check ttl (can I send this info to others ?)
      if ( (hdrip->ttl() < 1) ) {  // TTL expired,
                                   // don't forward
      //Drop the packet
      …
      }
      // The packet can live more
      // let's forward by broadcast
      // re-use packet, no need to drop

      pkt_send(p,IP_BROADCAST);
      break;
case IARP_DATA: // got a data packet, forward if I am
                // relay, or send up if I am
                // destination.
      // Method to consume the packet (If I am a
      // destination). And another method to relay
      // the packet if the destination is some other
      // node
      …
      …
```

```
                break;

            case IERP_REPLY: // got a reply, process as a relay
                             // or destination.
                …
                …
            …
            …

            break;

            case IERP_REQUEST: // outgoing request. QD,ET as a
                               // relay or bordercaster.
            …
            …
            …
            …


        default: // Unknown ZRP Packet Type

        exit

    }
}
```

The above method is to receive a ZRP packet type which is defined as

```
enum ZRPTYPE // Types of ZRP packets
{
  NDP_BEACON, NDP_BEACON_ACK, IARP_UPDATE, IARP_DATA,
  IERP_REPLY, IERP_REQUEST
};

// ZRP header structure
struct hdr_zrp {
  int zrptype_;
  int routeindex_;

  // IARP hdr
  int seq_;
  int numentries_; // number of link entries in this update
                   // or length of direct routing list
  nsaddr_t originator_; // for ndp, iarp or ierp
  nsaddr_t ierpsource_;
  nsaddr_t ierpdest_;
  nsaddr_t iarpsource_;
  nsaddr_t iarpdest_;
  nsaddr_t lastbordercaster_;
  Time pktsent_;
  int enc_dport_;
  int enc_daddr_;
  packet_t enc_ptype_;

  int qid_;  // query id counter
```

```
  nsaddr_t *route_; // pointer to route list data
  linkie *links_;  // pointer to link state list

  int forwarded_;  // TRUE if forwarded before
};
```

In addition to ZRPAgent class, there are also two other classes LinkStateTable and RoutingTable. The most important method for the LinkState class is the update() method. The input is a link_struct, which includes source, destination, and the linkstate. If this link is in the table, the state is copied to the linkstate entry stored in the table, the link expiry is updated and the sequence number of the incoming link state data is also copied to the stored linkstate entry. If the link is not found in the table, the link is added and stamped with an expiry time.

The RoutingTable class method compute_routes() computes a minimum spanning tree starting from the current node. The data structure is an array of linked lists. Each array element is an entry for a destination, and contains the destination's id in the data structure. The next hop from that node can be found by looking at its "next_" pointer, which just points to another array element. All entries, if iterated through as a linked list, end up finally at the first entry, which is the current node's route entry for itself.

Below is a structure of these two classes

```
class LinkStateTable {

      public:
            LinkStateTable() : lshead_(NULL) { } // Constructor
            void add(link_struct *link);  // Adding entry to the
                                          // table
            int remove(link_struct *link); // Removing an expired
                                           // route from the link

            // Method to update other nodes
            BOOLEAN update(nsaddr_t src, nsaddr_t dest,
                        int state, int seq, int *seq_match);

            int exist(link_struct *link); // Checking for the
                                          // existence of a link

  };

class RoutingTable {
      public:
            RoutingTable(ZRPAgent* a)  { // init
                num_entries_=(0);
                agent_ = a;
```

```
        }

        ZRPAgent* agent_;
        nsaddr_t  my_address_;
        int num_entries_;

        // Tables
        rtable_node* route[MAX_NODES_PER_ZONE+1];
        rtable_entry* outer_route[MAX_OUTER_ROUTES];

        rtable_node* periphnodes_;
        LinkStateTable *linkstatetable;

        // Methods
        void RoutingTable::compute_routes(nsaddr_t myadd, int
                                          radius);
        void erase();
        int get_entry(nsaddr_t node_id);
        int route_exists(nsaddr_t id);

        // drop from link table, add node to RouteTable
        void add_drop(link_struct* ls, int i);

        // Finds Next Hop On route to given,
        //returns FALSE if doesn't exist
        int next_hop_to(nsaddr_t node) { }

        // returns route entry index for given node
        int node_exists(nsaddr_t nodei);

        void empty_links(link_struct *lst);
};
```

Data structure for the routing table

```
// A single data struct naming a node, used for storing node
// addresses in various tables.
struct rtable_node {
  nsaddr_t id_; //Node id
  int numhops_; // number of hops from me
  rtable_node* next_;
};

// This struct is a head pointer for a route entry, has expiry
// timestamp. Route is at next_.
struct rtable_entry {
  nsaddr_t id_;  // Node id
  Time expiry_;  // route information expiry time
  int numhops_;  // number of hops from me
};
```

Here are some of the default variables from the Tcl simulation script that a user can change.

```
DEFAULT_BEACON_PERIOD 5 // period of beacon transmission in sec
DEFAULT_NEIGHBOR_TIMEOUT 15 //seconds
DEFAULT_NEIGHBOR_ACK_TIMEOUT 2 // sec
DEFAULT_NEIGHBORTABLE_SCAN_PERIOD 11 // scan table every n sec
DEFAULT_IARP_UPDATE_PERIOD 22 //time between periodic update(sec)
SUSPEND_FLAG FALSE // all in/outgoing packets are dropped if TRUE

MAX_ROUTE_LENGTH 1000
IERP_TTL 40

MAX_NUM_NEIGHBORS 20

IERP_REQUEST_TIMEOUT 5 // secs before timeout on an ierp request
OUTER_ROUTE_EXPIRATION 20 // secs that a cached route will last

HDR_BEACON_LEN 20 // bytes

IERPCACHE_SCAN_PERIOD 5
MAX_ROUTE_LENGTH 20
```

5.4. Installing ZRP on NS-2

Even though the current releases of NS-2 do not include ZRP implementation, a new addition to them is quite straight forward due to the object oriented nature of the NS-2 architecture and implementation. In order to add a new protocol, one needs to add (modify) some files from the NS-2 source code and do recompilation. The figure below shows the different packages of NS-2 source code and their architecture.

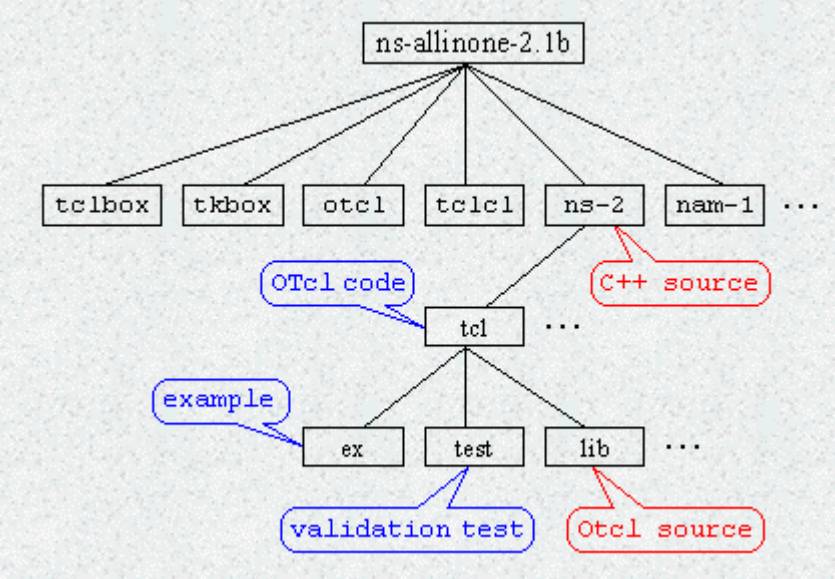


Fig 5.2 NS-2 directory structure

The first step (to do) is, to introduce the new protocol (ZRP) to NS-2. This can be done by editing file 'packet.h'. Under this file we will find the definitions for the packet protocol IDs (i.e. PT_TCP, PT_TELNET, PT_AODV, PT_DSR etc.). ZRP can be added to NS-2 by inserting **"PT_ZRP"** in last few lines of packet enumeration enum packet_t {} as shown below.

```
enum packet_t {
  PT_TCP,
  PT_UDP,
        ......
  // insert new packet types here
  PT_TFRC_ACK,
        PT_ZRP,     //  packet protocol new routing, ZRP
  PT_NTYPE // This MUST be the LAST one
```

```
};
```

There is also an additional change that needs to be made to the same source file **packet.h** under the section of **p_info()**

```
class p_info {
public:
        p_info() {
                name_[PT_TCP]= "tcp";
                name_[PT_UDP]= "udp";
                ..........
                name_[PT_RTPROTO_DV]= "rtProtoDV";
                name_[PT_CtrMcast_Encap]= "CtrMcast_Encap";
                name_[PT_CtrMcast_Decap]= "CtrMcast_Decap";
                name_[PT_SRM]= "SRM";

                name_[PT_ZRP]= "ZRP";

                name_[PT_REQUEST]= "sa_req";
                name_[PT_ACCEPT]= "sa_accept";
                name_[PT_CONFIRM]= "sa_conf";
                name_[PT_TEARDOWN]= "sa_teardown";
                name_[PT_LIVE]= "live";
                name_[PT_REJECT]= "sa_reject";
                name_[PT_NTYPE]= "undefined";
                ..........
        }
        .....
}
```

Libraries under Tcl also need to know about this new protocol so that during simulation a user can only specify some parameters like **Agent/ZRP.** These libraries need to know detail information about this new addition before they give service to the user. Information about ZRP can be given to the Tcl objects by doing to the following modifications.

tcl/lib/ns-agent.tcl – add ZRP information at the end of this file as shown below.

```
..........
Agent/AODV set sport_   0
Agent/AODV set dport_   0

Agent/ZRP instproc init args {
        $self next $args
}
Agent/ZRP set sport_   0
Agent/ZRP set dport_   0
```

tcl/lib/ns-default.tcl – modify this file as shown below

```
...........
Agent/RAP set dpthresh_ 50
Agent/RAP instproc done {} { }

# Routing protocol agents
Agent/Mcast/Control set packetSize_ 80

Agent/ZRP set packetSize_ 100


# Dynamic routing defaults
Agent/rtProto set preference_ 200  ;#global default preference
Agent/rtProto/Direct set preference_ 100
Agent/rtProto/DV set preference_      120
..........
..........
```

tcl/lib/ns-lib.tcl – looks like the following after modification

```
switch -exact $routingAgent_ {
    ..........
    ..........
    AODV {
            set ragent [$self create-aodv-agent $node]
    }
    TORA {
            Simulator set IMEPFlag_ ON
            set ragent [$self create-tora-agent $node]
    }
    ZRP  {
            set ragent [$self create-zrp-agent $node]
    }
    DIFFUSION/RATE {
            eval $node addr $args
            set ragent [$self create-diffusion-rate-agent $node]
    }
    ..........
    ..........
Simulator instproc create-aodv-agent { node } {
        #  Create AODV routing agent
        set ragent [new Agent/AODV [$node id]]
        $self at 0.0 "$ragent start"     ;# start HELLO Messages
        $node set ragent_ $ragent
        return $ragent
}

Simulator instproc create-zrp-agent { node } {
        set ragent [new Agent/ZRP [$node id]]
        $node set ragent_ $ragent
        $self at 0.0 "$ragent start"    ;# start updates
        return $ragent
}
..........
..........
```

tcl/lib/ns-packet.tcl

```
        ..........
        ..........
        foreach prot {
                ..........
                ..........
                AODV
                ARP
                UMP
                ZRP
                TORA
                Pushback
                NV
        } {
                add-packet-header $prot
        }
        ..........
        ..........
```

The trace modules have to also be modified so that we can get trace files while running ZRP simulations over NS-2. With all the above modifications, the NS-2 software can properly simulate the new ZRP routing protocol.

The trace file support of NS-2, which is an extension from CMU can be configured to trace ZRP packet during simulation by simply adding the method which is shown below to **cmu-trace.cc**.

```
Void CMUTrace::format_zrp(Packet *p, int offset){
        struct hdr_cmn *ch = HDR_CMN(p);
        struct hdr_ip *ih = HDR_IP(p);
        int src = Address::instance().get_nodeaddr(ih->saddr());
        int dst = Address::instance().get_nodeaddr(ih->daddr());
        if (newtrace_) {
            sprintf(pt_->buffer() + offset,
            "-Is %d.%d -Id %d.%d -It %s -Il %d -If %d -Ii %d -Iv %d ",
                    src,                            // packet src
                    ih->sport(),                    // src port
                    dst,                            // packet dest
                    ih->dport(),                    // dst port
                    packet_info.name(ch->ptype()),  // packet type
                    ch->size(),                      // packet size
                    ih->flowid(),                    // flow id
                    ch->uid(),                       // unique id
                    ih->ttl_);                       // ttl
        } else { // Old Trace format
          sprintf(pt_->buffer() + offset, "------- [%d:%d %d:%d %d %d] ",
                src, ih->sport(),
                dst, ih->dport(),
                ih->ttl_, (ch->next_hop_ < 0) ? 0 : ch->next_hop_);
        }
}
```

Since cmu-race.cc file defines some parameters on its header file **cmu-trace.h**, it is also required to define our ZRP addition here as shown below.

```
..........
..........
#define DROP_RTR_IERP   "IERP"
#define DROP_RTR_IARP   "IARP"
#define MAX_ID_LEN     3
#define MAX_NODE       4096
..........
..........
void    format_arp(Packet *p, int offset);
void    format_dsr(Packet *p, int offset);
void    format_zrp(Packet *p, int offset);
..........
..........
```

Since all the above changes are made on the source codes of NS-2, we need to recompile the entire NS-2 software so that our changes can take effect. NS-2 can recompile our new additions and modification if we use a command **"make clean"**, followed by "**make**" from the NS-2 directory. Since the Linux command make is used to read configuration parameters and values defined under Makefile, we need to inform the compiler by modifying Makefile as shown below.

```
dsdv/dsdv.o dsdv/rtable.o queue/rtqueue.o \
zrp/zrp.o zrp/zrp_table.o \
routing/rttable.o \
```

The compiler will fail to compile our ZRP code, if we don't place it in the same folder with the NS-2 c++ source files.

6. Simulation

6.1. Simulation overview

A typical simulation with NS-2 is shown in figure 6.1. The user has to generate the following files which are inputs to Ns-2 simulation software.

- A scenario file that describes the movement pattern of a node
- A communication file that describes the traffic in the network.

These files can be generated by drawing by hand using the visualization tool Ad-hockey [19], or by generating a completely randomized movement and communication patterns with a script.

These files are then used for the simulation and as a result a trace file is generated as an output. Prior to simulation, the parameters that are going to be traced during the simulation must be selected. The trace file can be scanned and analyzed for various parameters that we want to measure. This can be used as data for plots with for instance Gnuplot plotter (that the research has used). The trace file can be used to visualize the simulation run with the Network Animator (NAM).

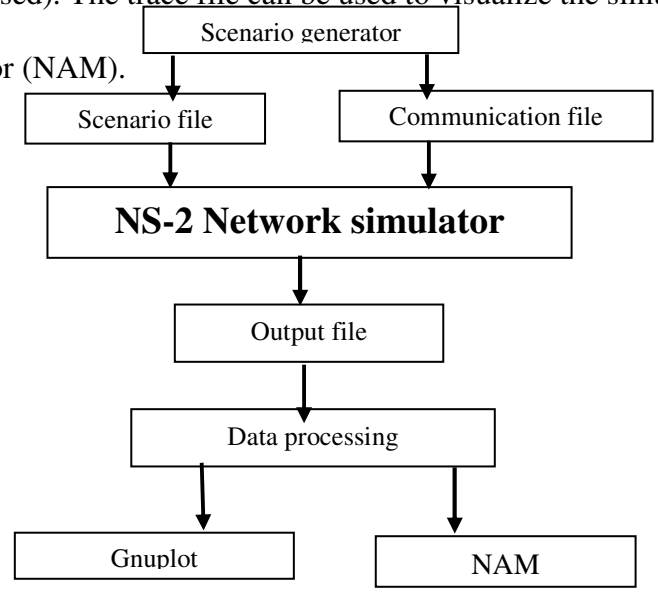


Fig 6.1 Simulation overview in NS-2

6.2. Performance Metrics

There are several quantitative metrics that might be used to analyze the performance of a routing protocol. The RFC 2501[10] defines the following measures. Basically they can be used to analyze the performance of any routing protocol.

The following measurements are defined in RFC 2501

1. End-to-end data throughput and delay

   This metric involves analyzing the efficiency of data routing. Statistical measures (e.g. means, variances and distributions) are essential. These are used to analyze the effectiveness of a routing policy as measured from the external perspective (the perspective of other policies that utilize routing).

2. Route acquisition time

   This, measures the time required to establish routes. It is an end-to-end measurement. Route acquisition time is concerned especially with on-demand routing approaches.

3. Percentage Out-of-Order Delivery

   This measures the connectionless routing performance. It's an interesting metric especially from the transportation layer's point of view (e.g. TCP) since Transportation layer prefers mostly in-order delivery.

4. Efficiency

   This refers to internal effectiveness of a routing policy. Thus, to achieve a certain externally evaluated data routing efficiency, two policies may consume different amounts of overhead since their internal efficiencies differ. If control and data traffic use the same transmission channel, then excessive control traffic will probably affect the internal efficiency of a policy.

While analyzing the performance of ad hoc routing protocols, some parameters of the networking context should be considered carefully. The RFC 2501 [10] defines the following ones:

1. Network size

   This is simply measured by the number of nodes in a network.

2. Network connectivity

   This refers to the average number of neighbors of a node.

3. Topological rate of change

   This shows the rate of change of network's topology.

4. Link capacity

   The effective speed of a link, which is measured in bits per second

5. Fraction of unidirectional links

   This must be concerned when evaluating, how the number of unidirectional links present in a network affect the protocol's performance.

6. Traffic patterns

   This is concerned when evaluating how well a protocol can adapt to non-uniform or bursty traffic patterns.

7. Mobility

   This is concerned when assessing how relevant temporal or spatial topological correlation is to the performance of a routing protocol.

8. Fraction and frequency of sleeping nodes

   How well a protocol can handle situations where sleeping nodes are present?

6.3. Simulation model

In this thesis, the performance metrics that are used to evaluate and compare the routing algorithms are:

1. **Packet delivery fraction** – The ratio of data packets delivered to the destinations to those generated by the source.

2. **Average end-to-end delay of packets** – This includes all possible delays caused by buffering during route discovery latency, queuing at the interface queue, retransmission delays at the MAC, and propagation and transfer times.

3. **Routing overhead** – The number of routing packets transmitted per data packet delivered at the destination. Each hop-wise transmission of a routing packet is counted as one transmission

Simulations are done based on the mobility that is mobility simulation setup, traffic or load capability simulation setup and network size simulation set up.

6.3.1. Mobility Simulation setup

Nodes in this simulation move according to a model that is called the "random waypoint" model [20]. The movement scenario files that are used for each simulation are characterized by a pause time. Each node begins the simulation by remaining stationary for pause time seconds.

It then selects a random destination in the 500×500 rectangular area and moves to that destination at a speed distributed uniformly between 0 and some maximum speed (20m/sec). Upon reaching the destination, the node pauses again for pause time seconds, selects another destination, and proceeds there as previously described, repeating this behavior for the duration of the simulation

The node-movement generator is available under ~ns/indep-utils/cmu-scen-gen/setdest directory and consists of setdest{.cc,.h} and Makefile.[19] The command would look like

```
./setdest [-n num_of_nodes] [-p pausetime] [-s maxspeed] [-t simtime] \ [-x maxx] [-y maxy] > [outdir/movement-file]
```

The scenario that is used for this simulation is shown in the following table.

Number of Nodes	50
Simulation time	100sec
Number of connections	10
Maximum Speed	20m/s
Sending rate	8
Network area	500×500
Traffic type	CBR
Pause time	0,10,20,30,….90
Agent	UDP
Packet size	512bytes

Table 6.1 Scenario of movement simulation

Using the above scenario, comparisons of packet delivery fraction, Routing overhead and End-to-end delay based on pause time of the following routing algorithms:
DSDV, DSR, TORA AODV and ZRP.

6.3.1.1. Results and Analysis

The following section discusses the simulation results for mobility simulation and compares with the actual algorithm of each routing protocols

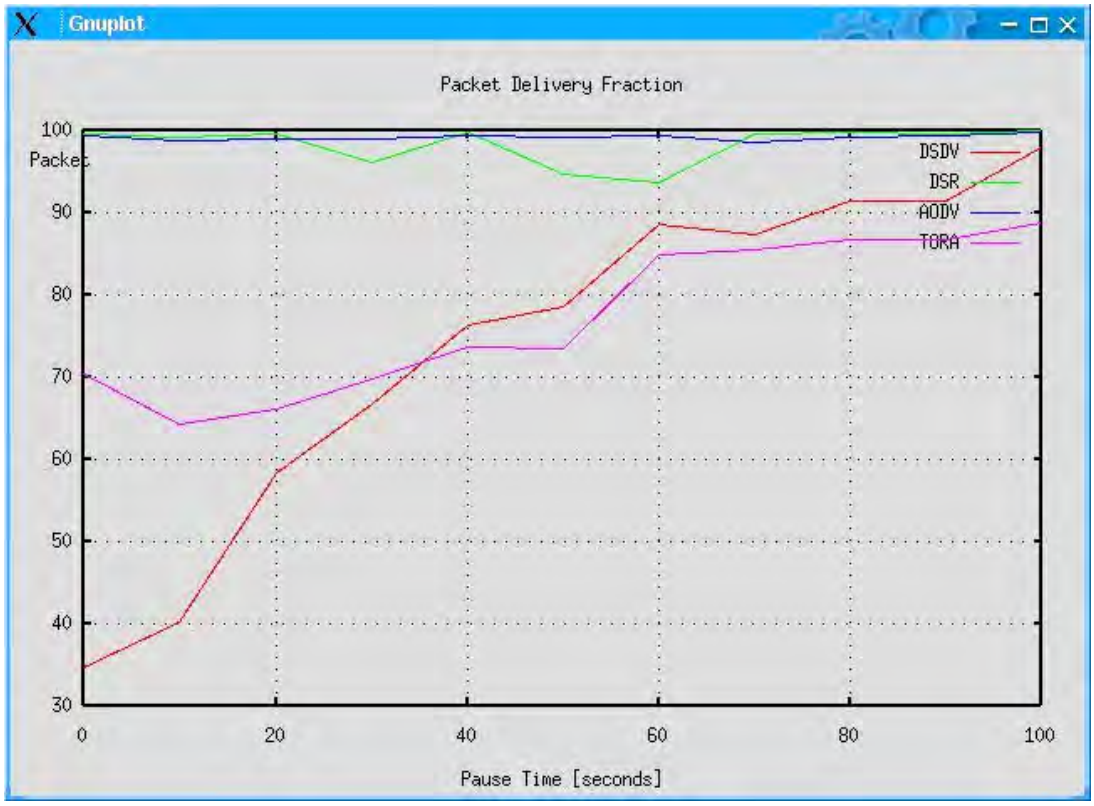


Fig 6.2 Packet delivery fraction Vs. Pause time

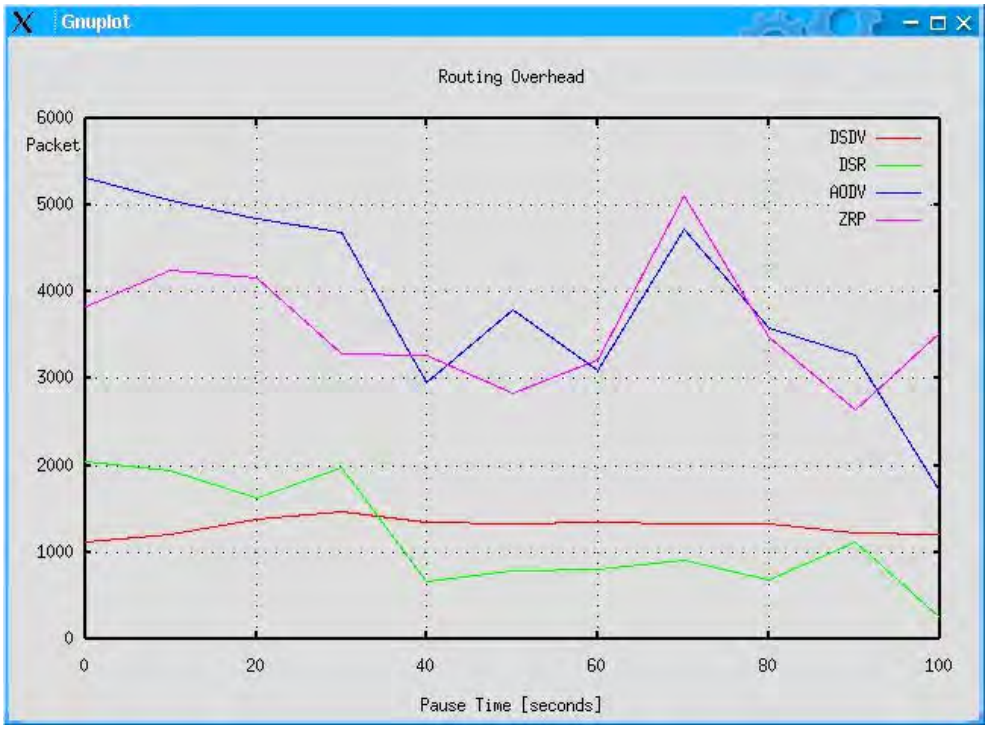


Fig 6.3 Routing packets Vs. Pause time

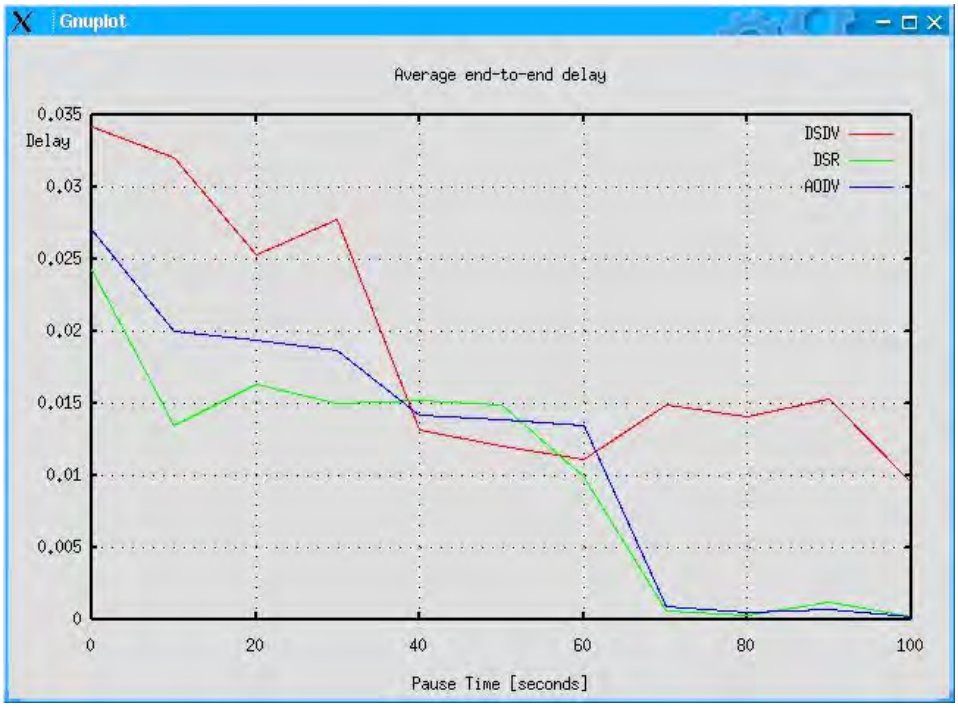


Fig 6.4 Average End-to-end delay Vs. Pause time

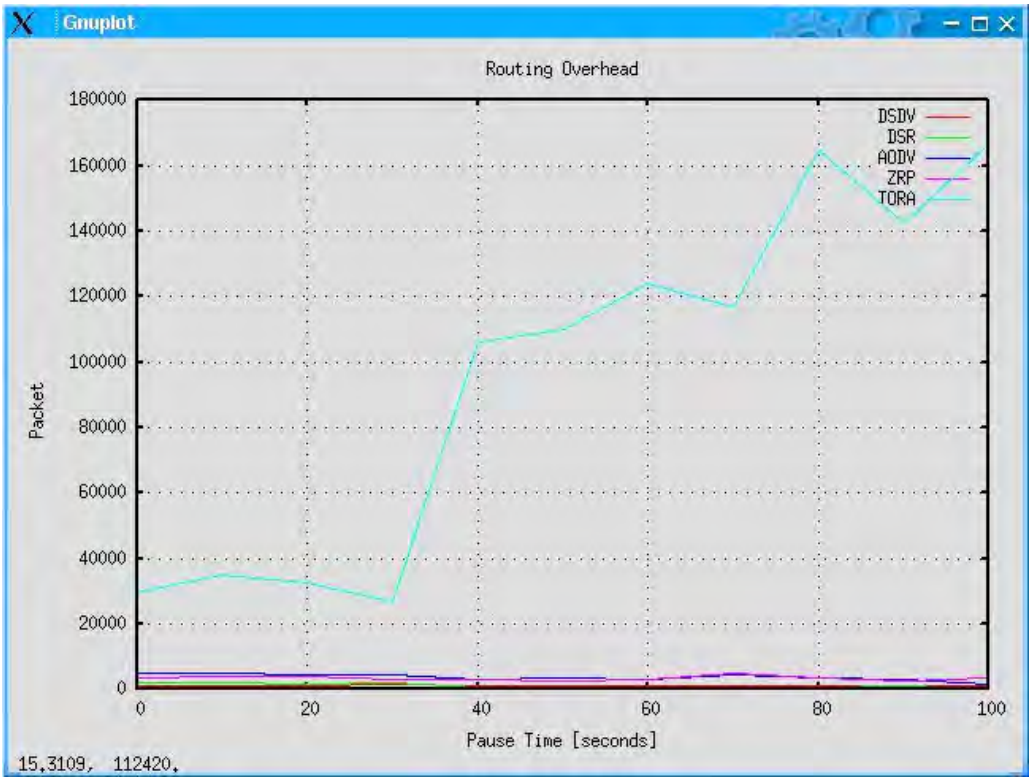


Fig 6.5 Routing packets Vs. Mobility (including TORA)

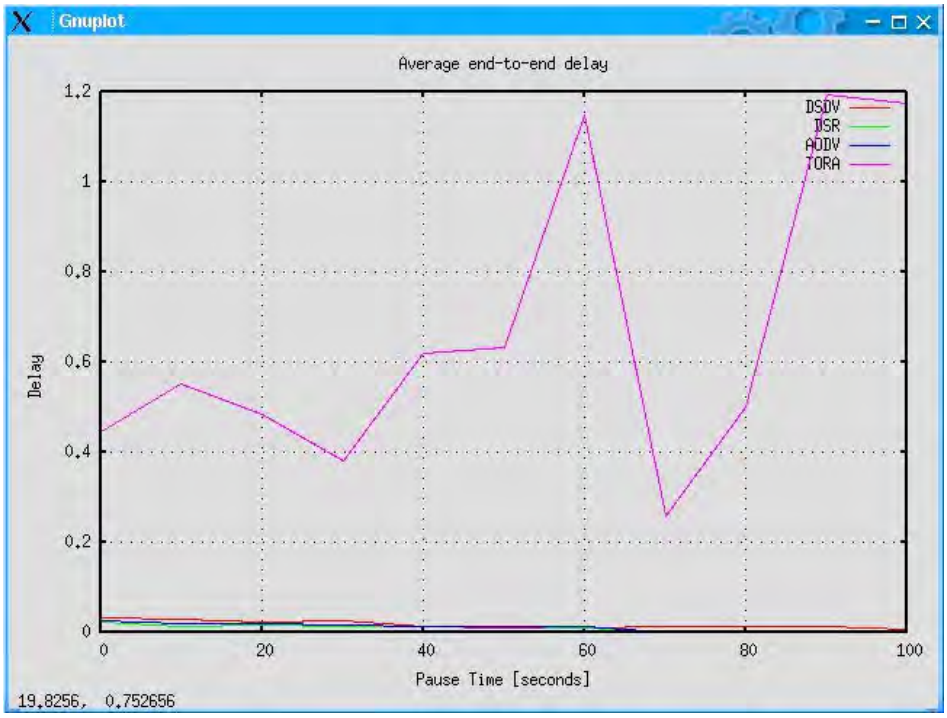


Fig 6.6 Average End-to-end delay Vs. Pause time (Including TORA)

Looking at the above fig6.2, the ad-hoc routing protocol DSDV fails to converge at high mobility. This is because DSDV is a proactive distance vector which maintains routing table and this routing table is to be broadcasted periodically and on-demand basis. At high mobility, updating becomes fast and full dump; that is all the routing table contents are to be sent by each node. The routing table consists of the following information

- All available destinations
- Next hop for each destination
- Number of hops to each available destination
- A sequence number for each route table entry, originated by the destination station

Broadcasting such information frequently creates a highly congested media and it is difficult to maintain consistent information on every node. In general, DSDV suffers the following problems:

- maximum settling time is difficult to determine
- does not support multi-path routing
- destination centered synchronization suffers from latency problem
- Excessive communication overhead due to periodic and triggered update.

Routing protocols DSR and AODV behave well because of their reactive nature as shown in the above figures. At high mobility the two routing protocols can deliver more than 95% of the sent packets. When we compare the two routing protocols:

DSR is a source routing as discussed in chapter 3. In every packet the route information starting from the source to the destination is included as header information. This might make the size of the packet very large, and may create congestion as compared to hop-to-hop basis routing which adds only the next node address. However, this will decrease the load incurred at each node for adding header information.

AODV is also a reactive distance vector routing protocol that maintains a routing table containing the following information:

- Destination address

- Next hop address
- Sequence number
- List of the precursor nodes

It uses a hop-to-hop routing bases and senses neighbor links on regular bases using hello messages.

TORA is highly adaptive, loop-free, distributed routing algorithm based on the concept of link reversal. It was proposed to operate in a highly dynamic mobile networking environment. It is source initiated and provides multiple routes for any desired source/ destination pair. This algorithm requires the need for synchronized clocks. Its basic functions of the protocol, namely route creation, route maintenance and route erasure.

At high mobility, TORA delivers less than DSR and AODV because it uses inter-nodal co-ordination and it also exhibits instability behavior similar to "count-to-infinity" problem in distance vector routing protocols. There is a potential for oscillations to occur, especially when multiple sets of coordinating nodes are concurrently detecting partitions, erasing routes, and building new routes based on each other. This instability is considerable as mobility increases.

Looking at the routing overhead incurred by each routing algorithm in fig. 6.3, as the mobility increases the routing packets are also increases. TORA has high routing overhead as compared to others and it couldn't be drawn its plot on the same scale with others in fig.6.3. Routing overhead of TORA is shown in fig6.5, as you can see; it is much considerable as compared to others.

DSDV has almost constant overhead through various node mobility as shown in the fig6.3. This shows that DSDV is a proactive distance vector and updating is taking place not only when there is a link status change but also periodically.

DSR has the least overhead as compared to others. This is because of its source routing nature but as the size of the network increases, its routing overhead is considerable.

AODV has more routing overhead than DSR because in addition to RREQ and RREP messages, it checks neighbor links by sending hello messages and it is also a distance vector that maintains routing tables.

The new hybrid routing protocol ZRP behaves well as compared to purely reactive or purely proactive because the focus of ZRP is to solve scalability problems of these protocols. At high mobility, the reactive nature of ZRP increases and more routing overhead is expected. More discussion about ZRP is given below.

When looking at the delay, DSR and AODV work well even at high mobility. However, the delay of DSDV is considerable at high mobility since convergence is difficult at highly mobile nodes.

TORA has a considerable delay as compared to all other routing protocols especially as the mobility increased as shown in fig6.6; this is the reason that is not included in fig6.4 using the same scale. The delay of TORA is mainly its algorithmic nature it takes considerable time in route creation using the link reversal algorithm.

6.3.2.Network Offered Load Simulation

This simulation is conducted by varying the number of traffic sources (or number of connections) to evaluate the load capability of each routing algorithm.

The following table shows the scenario that are used for this simulation

Number of Nodes	100
Simulation time	50sec
Maximum Speed	20m/s
Network area	500×500
Pause time	10sec
Traffic type	CBR
Agent	UDP
Sending rate	4
Packet size	512bytes
Maximum connection	10,20,30…..90

Table6.2 scenario of network offered load simulation

I have plotted packet delivery fraction, Routing overhead and End-to-end delay based on the number of connections of the routing algorithms.

6.3.2.1. Results and Analysis

This section shows the load capability of each algorithm based on the simulation results.

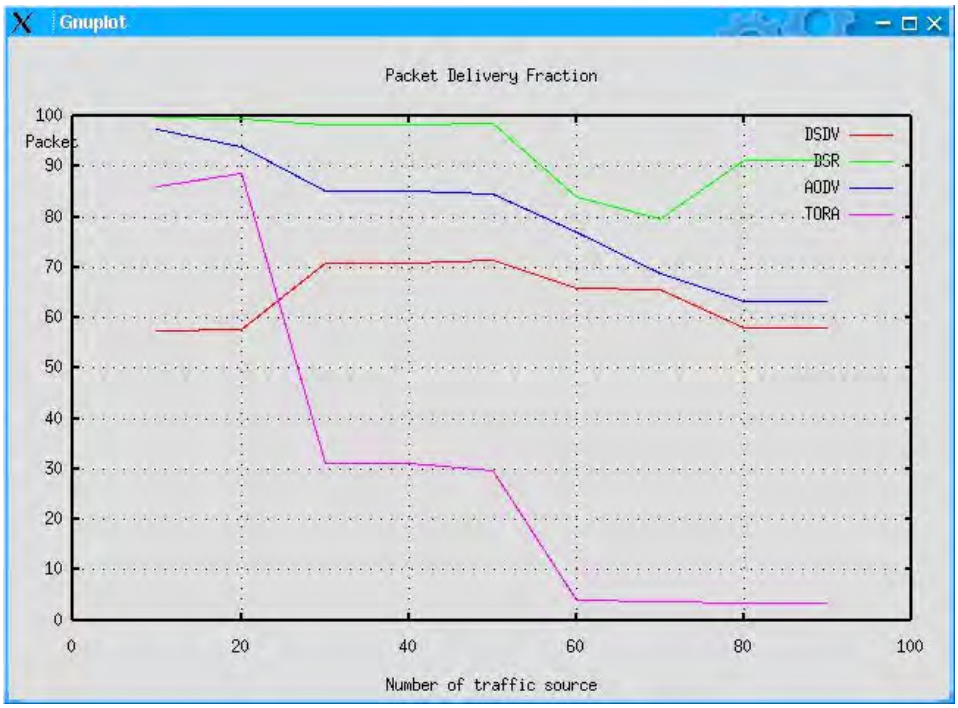


Fig6.7 Packet delivery fraction vs. Number of traffic source

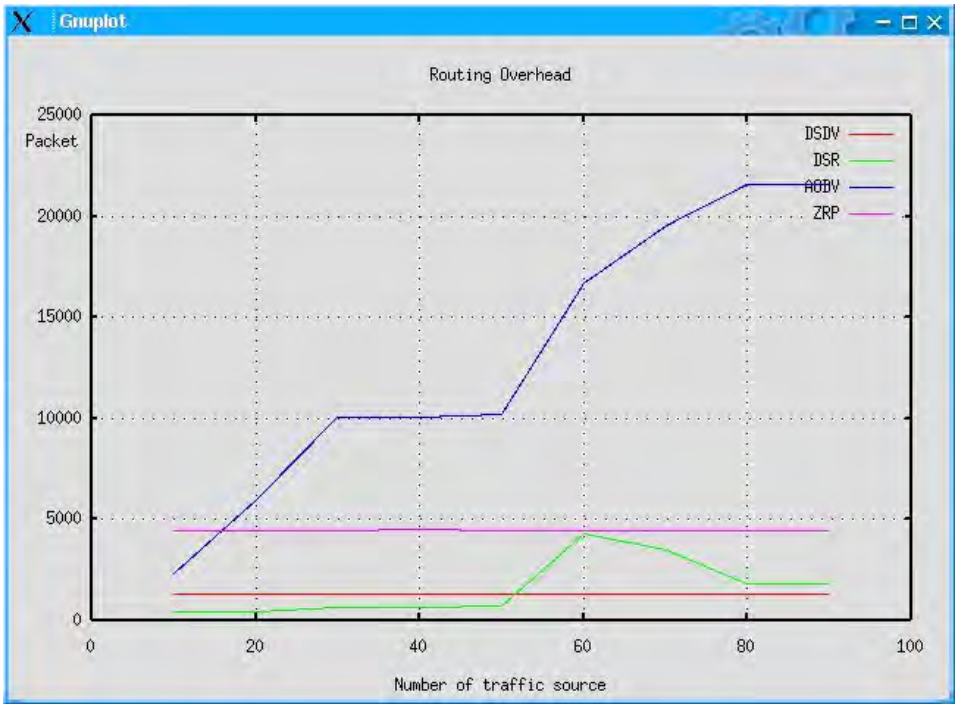


Fig6.8 Routing packets vs. Number of traffic source

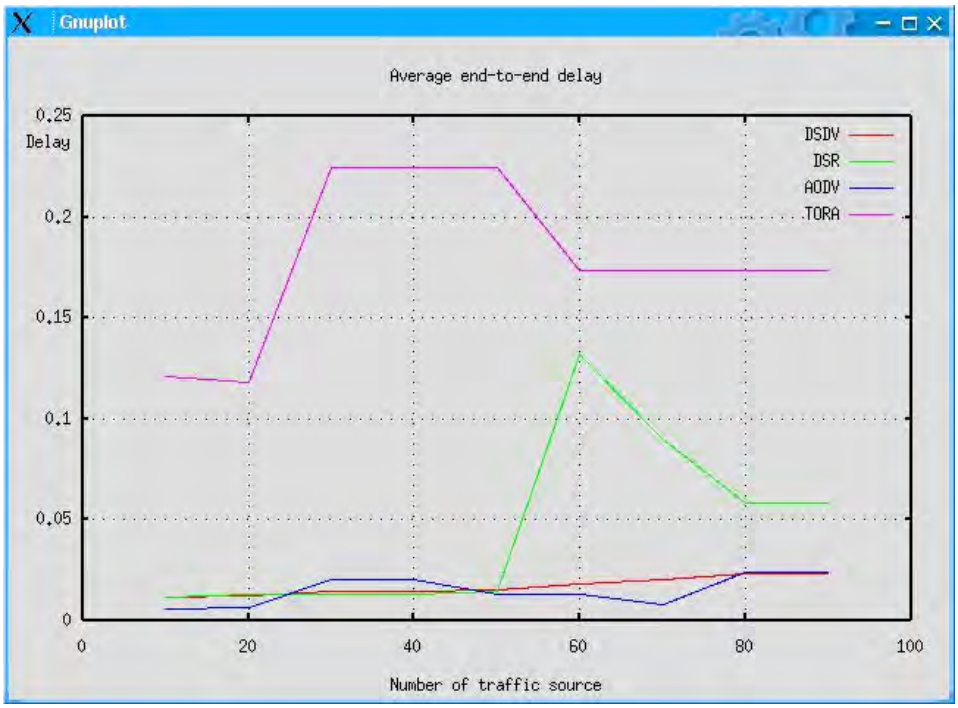


Fig6.9 Average end-to-end delay vs. Number of traffic source

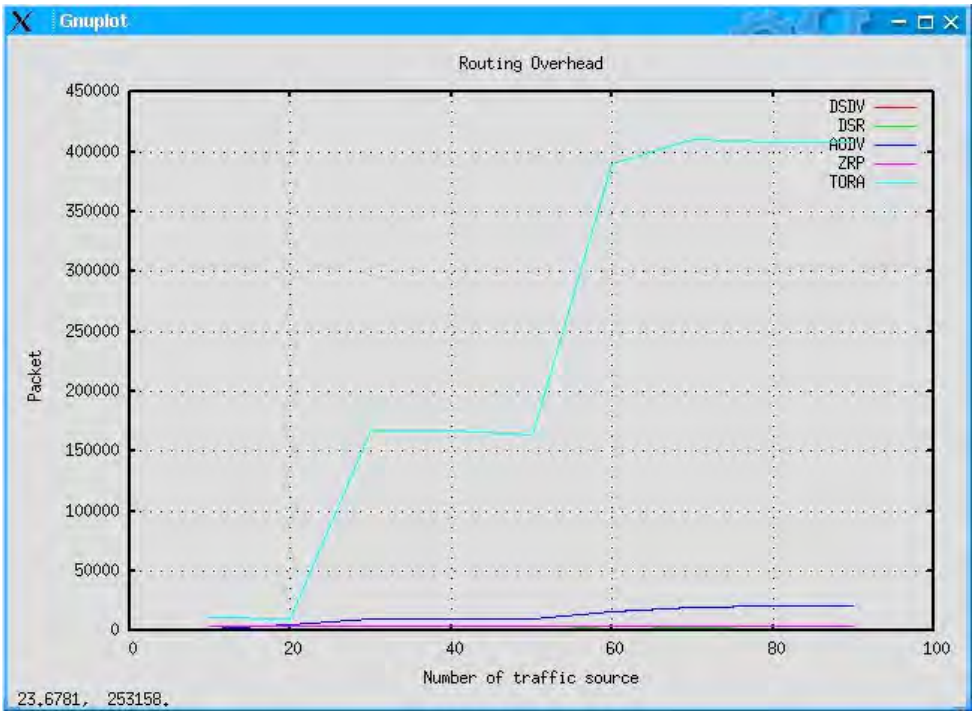


Fig 6.10 Routing Packets Vs. Number of traffic sources (including TORA)

TORA does well with 10 or 20 sources, delivering between 85% to 90% of sent data packets as shown in the fig6.7. As the sources increased TORA's average packet delivery ratio dropped to 30%, which is a drastic drop. The above scenarios show that TORA fails to converge because of increased congestion, due to high routing overhead.

DSR and AODV deliver more than 95% at less number of sources as shown in the figure, but as the number of sources increased they tend to decrease the delivering capability.

DSDV has lower delivering capability than DSR and AODV because of its proactive nature.

The nature of DSR and AODV is similar as both discover routes on-demand basis. But the routing overhead exhibited as the number of sources increased is absolutely different as shown in fig6.8. As the number of sources increased the RREQ packets of AODV is increased to discover the intended destinations. Unlike AODV, DSR uses its route cache instead of broadcasting RREQ messages.

TORA's overhead is quite large as compared to others and it is drawn in a separate scale as shown in fig6.10.

Looking at ZRP, as the number of sources increased it exhibited an almost constant routing overhead. This is due to BRP of the ZRP protocol, which considerably decreases the required RREQ and RREP packets as discussed above.

As the number of sources increased, the delay should also have increased because every source tries to send information. Since we have shared media, it becomes congested and that forces packets to be dropped. Fig6.9 tries to show this behavior. Unfortunately the plot is not clear enough, and this could be due to the simulation limitations.

6.3.3.Network Size Simulation

These simulations were carried on to determine the scalability nature of the routing algorithms. The goal was to determine how the algorithms behave as the number of nodes increased.

The simulations were conducted to create more realistic scenarios by varying the number of nodes and corresponding number of traffic sources. For example, if the number of nodes in the network is 80, then 30% of these nodes (24 nodes) were chosen as traffic sources.

The following table shows the scenario used for this simulation

Simulation time	50sec
Pause time	10sec
Sending rate	4
Network area	500×500
Traffic type	CBR
Agent	UDP
Maximum speed	20m/s
Packet size	512bytes
Number nodes	40,80,120,…200
Corresponding number of connections	12,24,36,…...60

Table6.3 Scenario of network size simulation

Like the above simulations, the packet delivery fraction, Routing overhead and End-to-end delay are plotted based on the number nodes (network size) of the routing algorithms.

6.3.3.1. Results and Analysis

This section shows the scalability nature of each routing algorithms based on the simulation results.

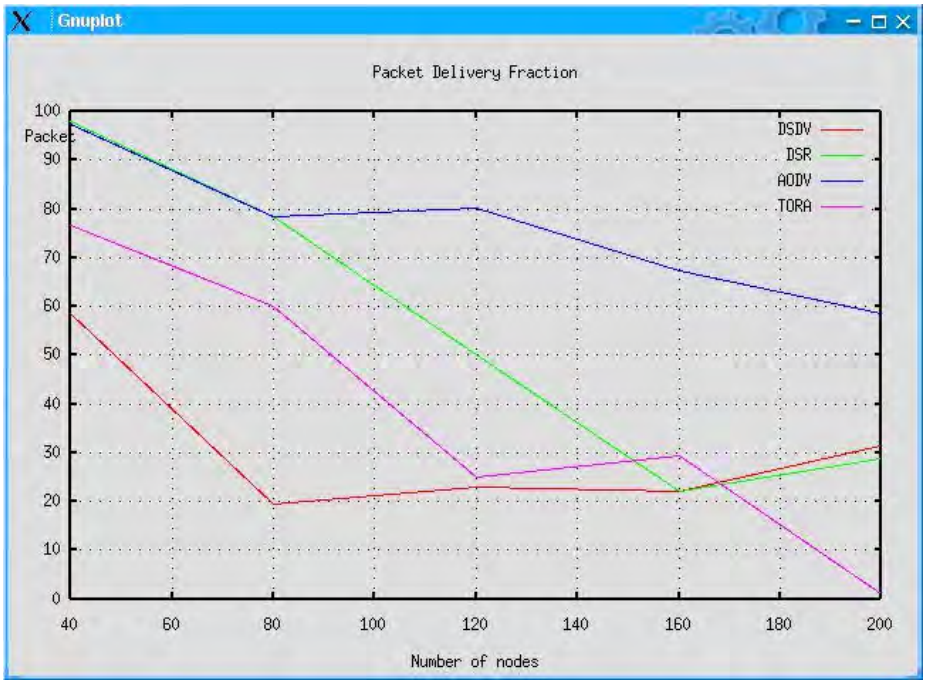


Fig 6.11 Packet delivery fraction vs. Number of nodes

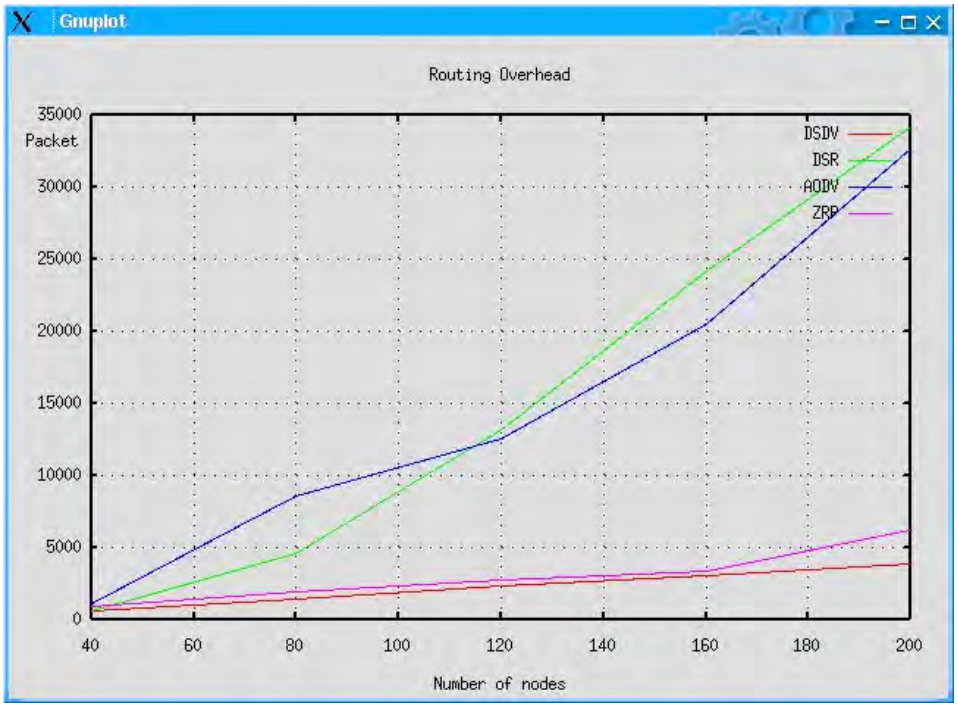


Fig 6.12 Routing packets vs. Number of nodes

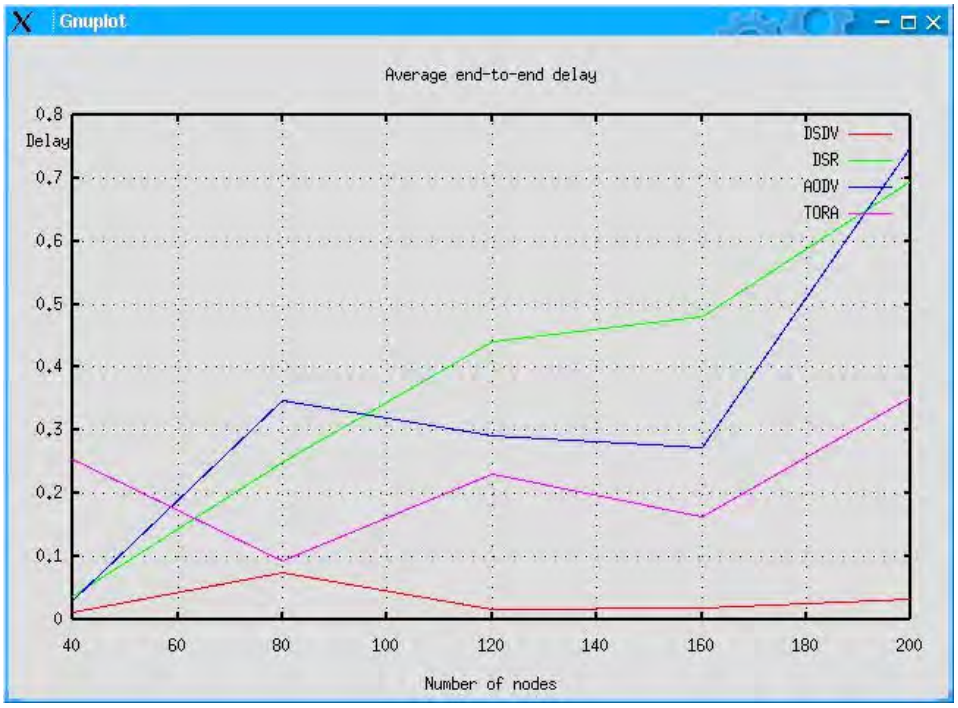


Fig6.13 Average end-to-end delay vs. Number of nodes

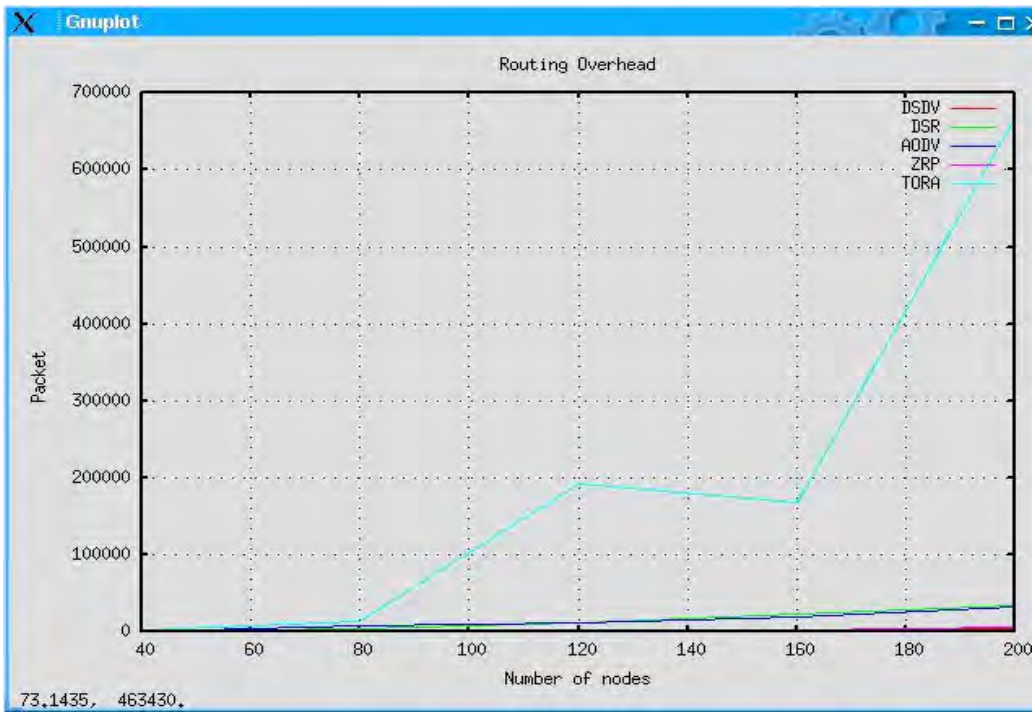


Fig 6.14 Routing Packets Vs. Number of nodes (including TORA)

The above figures show the scalability nature of the routing algorithms. As expected, purely reactive and purely proactive protocols do not scale up very well as the network size is increased.

As shown above, DSR and AODV performed well at low number of nodes. As the number of nodes increased, the number of RREQ and RREP packets increased. This creates a congested media that drastically decreased their performance, especially for DSR as shown in the fig.6.11. The performance of DSDV also decreased as the network size increased.

The same explanation can be taken for fig6.12 and fig6.13.

TORA is one of the largest and most complicated reactive protocols. In terms of memory requirements, each node must maintain a structure describing the node's height as well as the status of all connected links per connection supported by the network as discussed above in chapter 3. In terms of CPU and bandwidth requirements, each node must be in constant coordination with neighboring nodes in order to detect topology changes and converge. As in the above simulation set up, TORAs overhead is much considerable than the others and it is drawn in separate scale as shown in fig6.14.

ZRP as shown in fig6.12 performs well as the size of the network increases. The driving factor for the proposal of ZRP is to make the ad-hoc routing protocols scalable, and the aim of this paper to show that ZRP alleviates the problem of scalability.

6.3.4.ZRP

One of the aims of the thesis work is to show that ZRP is the best ad-hoc routing algorithm as the size of the network is increased. This is due to the hybrid nature of the protocol that takes advantage of the two approaches. ZRP combines two radically different schemes of routing into one protocol. IntrAzone routing uses a proactive protocol to maintain up-to-date routing information to all nodes within the zone. By contrast, intErzone route discovery is based on a reactive route request/route reply scheme.

Several factors can influence the amount of control traffic or routing packets in ZRP. These include the routing zone radius, the network area or network span, and the density and velocity of nodes in the network.

The amount of proactive IARP routing packets increases with the routing zone radius. Because each node has to construct the bordercast trees for its interior nodes, a routing zone with a radius of ρ hops corresponds to an exchange of link-state information over a range of $(2\rho - 1)$ hops. For an unbounded network with a uniform distribution of nodes, the amount of intrazone control traffic is expected to be $O(\rho^2)$ [17]. However, in a network of finite size, the boundary effect makes the dependence less than ρ^2. In the case where ρ=1, there is no intrazone control traffic, because both the routing zone and the extended routing zone have a one-hop radius, and both consist solely of neighbor nodes. Therefore, all the information required to maintain connectivity within the routing zone is provided by the NDP.

On the other hand, the amount of reactive IERP control traffic produced by each route query decreases as the routing zone radius increases. When ρ=1, the neighbor nodes are the peripheral nodes. As a result, the IERP route query operates as a basic flood search. As the routing zone radius is increased, the route discovery procedure operates more efficiently, with a smaller amount of control traffic. This improved efficiency is a consequence of the query control mechanisms, namely query detection and early termination as discussed above, which combine to guide the route, requests outwards from previously queried regions.

The total amount of ZRP control traffic depends on network density as well as node mobility. An increase in network density has more of an impact on the proactive intrAzone route updates, due to a potential increase in the number of nodes in each routing zone. IntErzone route query traffic could also increase as a result of high network density. A dense network, consisting of a few nodes with high mobility, favors smaller routing zones because of the high cost in routing zone maintenance. Reactive configurations are therefore more appropriate in such a network environment. In contrast, a sparse network consisting of many nodes with low mobility would favor a more proactive configuration, as the cost of

maintaining larger routing zones is justified by the resulting reduction in route discovery traffic.

The span of the network does not affect the amount of intrAzone traffic. However, the amount of intErzone route query traffic increases as the network span increases. This is due to the increased number of route requests generated by the route discovery procedure. A large network span usually favors a large routing zone radius in order to reduce the amount of route query traffic, but, as we have seen earlier, this result in an increase in the amount of intrAzone control traffic instead. We will have to consider the tradeoffs between increased intrAzone control traffic and increased route query traffic in order to determine an optimal routing zone radius for a particular network.

However, it is difficult to compute the optimal zone radius because it depends on a number of factors, and these vary for different networks. These factors also vary as a function time even within a network. In addition, although it is possible to estimate node density, relative node velocity and network size, the performance of the protocol also depends on factors such as route selection criteria and data traffic behavior. A zone-sizing scheme has thus been proposed to find an appropriate zone radius depending on the ratio of IERP and IERP traffic. When the routing zone radius is less than the optimal zone radius, and the corresponding amount of ZRP control traffic is more than optimal, the traffic is dominated by IERP queries. If, however, the routing zone radius is larger than the optimal, and the traffic is more than optimal, then IARP route updates contribute to most of the traffic. In this scheme, the zone radius is increased if the IERP/IARP ratio is larger than a predefined threshold, and reduced if it is less than the threshold. The following figures shows IERP and IARP packets as a function of zone radius.

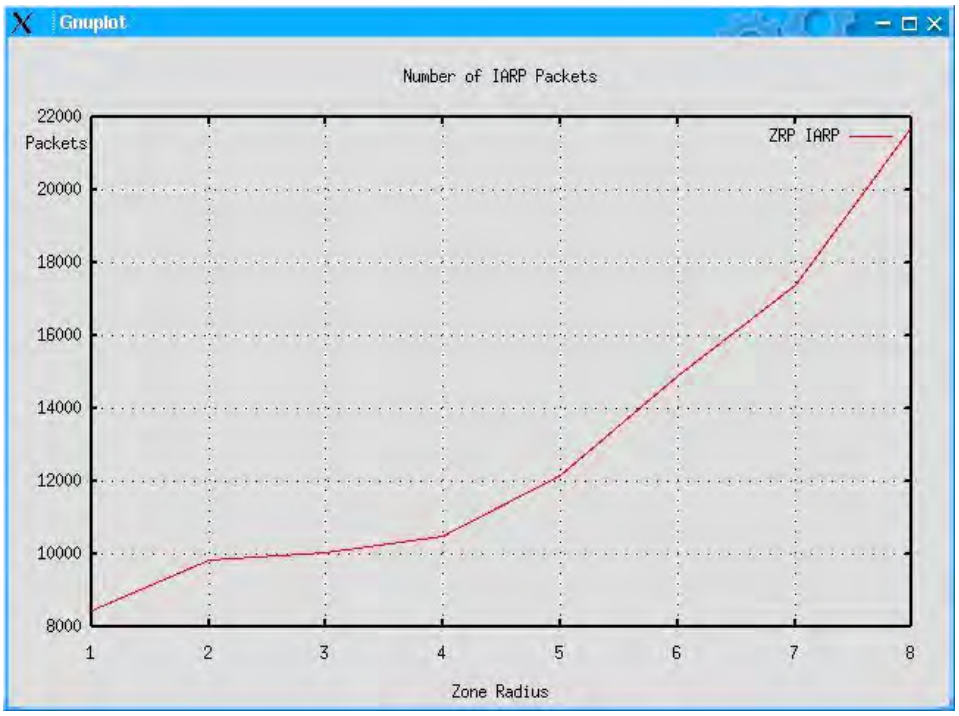


Fig 6.15 Number of IARP packets vs. Zone Radius

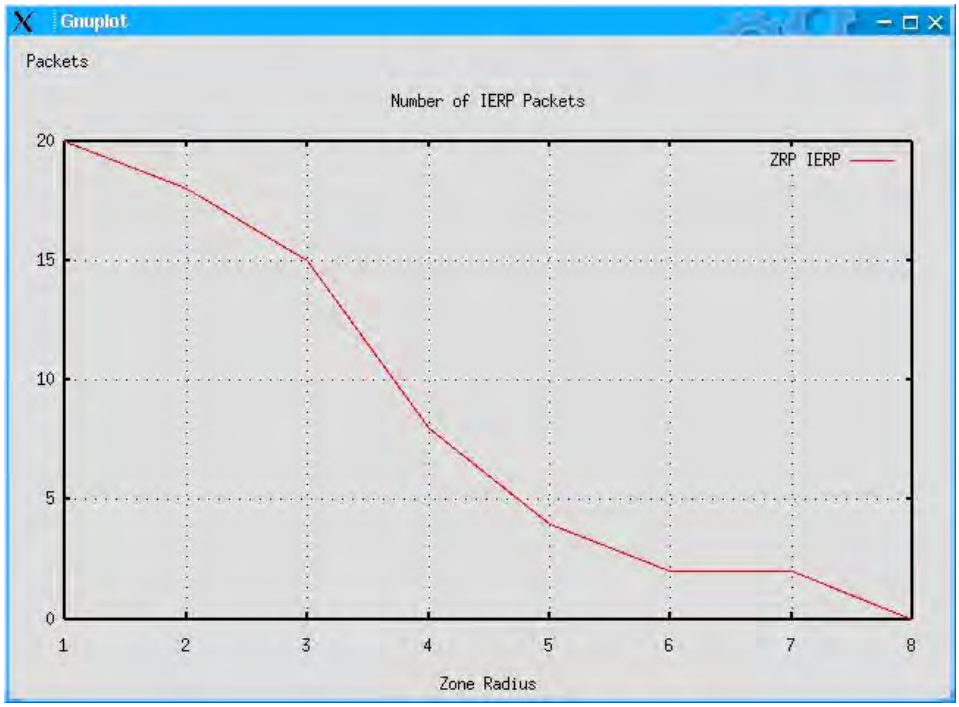


Fig 6.16 Number of IERP packets vs. Zone Radius

As can be seen from the above figures the amount of IARP control packets increased as the zone radius increased, and if the zone radius reduced the IERP control packets also increased. To make ZRP more efficient we can use efficient proactive protocols like OLSR for intrAzone routing and efficient reactive protocols like DSR for intErzone routing and we have to choose the optimized zone radius.

6.3.5.OLSR

One of the goals of this thesis work is to implement and simulate OLSR protocol and to make the necessary comparisons like the others. As can be seen from the routing algorithm stated in [5], it is difficult to interpret and change it to a runnable NS-2 code within this specified limit of time. Although I haven't put an actual plot, OLSR is an efficient proactive protocol proposed for large dense networks.

OLSR is a flat, distributed routing protocol. The proactive characteristic of the protocol implies that the protocol has all the routing information for all participating hosts in the network. However, the drawback is that the OLSR protocol needs that each host

periodically send the updated topology information throughout the entire network, which increases the protocol's bandwidth usage. However, the flooding is minimized by the MPRs, which are only allowed to forward the topological messages.

The reactiveness to the topological changes can be adjusted by changing the time interval for broadcasting the Hello messages. It increases the protocol's suitability for ad hoc network with the rapid changes of the source and destinations pairs. In addition, the OLSR protocol does not require that the link is reliable for the control messages, since the messages are sent periodically and the delivery does not have to be sequential. [5]

OLSR protocol is well suited for applications that do not require the long delays in the transmission of the data packets. The best working environment for the OLSR protocol is a dense network, where the most communication is concentrated between a large number of nodes. [5]

7. Discussion, Conclusion and Recommendation

7.1. Discussion

The area of ad hoc networking has been receiving an increasing attention among researchers in recent years, as the available wireless networking and mobile computing hardware bases are now capable of supporting the promise of this technology. With this advent of wireless networking, especially ad-hoc networking, issues of the traditional wired networking, such as routing, could not directly be applied on to this new scheme of communication.

Over the past few years, a variety of new routing protocols targeted specifically at the ad hoc networking environment have been proposed. An IETF workgroup called MANET, is currently working on mobile ad-hoc routing protocols and has set some desirable characteristics that any new routing algorithm should consider based on the nature of ad-hoc networks.

Generally the available proposed routing algorithms fall into three main categories, proactive, reactive and hybrid.

In proactive protocols, nodes store route information even before it is needed, by keeping track of routes for all destinations in the network. Such protocols are also known as table-driven protocols because of the use of routing tables. The advantage of proactive protocols is that a route can be immediately selected from the routing table when an application starts. As a result, packets can be sent without any delay.

On the other hand, additional control traffic is needed to update stale route entries repeatedly. This creates high congestion, consume the precious bandwidth and it may not be able to converge at high mobility as well as the size of the network becomes bigger and bigger. DSDV, OLSR, GSR, FSR and CGSR use this approach.

Reactive protocols obtain routing information when needed, by calculating route/paths only before the data transmissions takes place. This scheme eliminates the need for conventional routing tables at the nodes and the constant updating that has to be carried out when topology changes. Hence, lower bandwidth is used for the maintenance of the routing tables. On the other hand, these protocols have higher latencies because a route to the destination has to be acquired first before the data can be transmitted and as the network size increases the RREQ/RREP messages increase and degrade the network performance. AODV, DSR, TORA and CBRP use this approach.

Both the proactive and reactive protocols work well for small number of nodes. As the number of nodes increases in larger networks, zones or clusters of nodes are formed and hybrid protocols are used for better performance. Hybrid protocols attempt to assimilate the advantages of purely proactive and reactive protocols. Two different routing protocols are used, one between the nodes within the zone and another between the zones. Locality property holds since nodes tend to move solely within a zone and only a few moves between zones. Information on topology changes will only need to be made known to the nodes in the same zone. Nodes in other zone only need to know the route to the destination zone. As a result, local updates are performed more frequently within a zone compared to intErzone updates. ZRP uses this approach.

7.2. Conclusion

From the set of currently proposed MANET routing protocols, quite few are selected and analyzed in this thesis. The selection is in such a way that each selected routing protocol can represent the main categories of MANET routing group: proactive, reactive and hybrid.

Simulation results of this thesis work shows that DSR and AODV are the two routing protocols which performed well as they deliver more than 95% of the given packets to the destination. But these protocols are to suffer with delay as the nodes become highly mobile. This is because, both DSR and AODV are reactive and hence they need to first establish a route in order to send a packet at hand. Though DSR uses route caching, its cache becomes stale and more packets are dropped. In the case of AODV, high mobility requires a larger number of routing packets, which in turn increases the routing overhead. This is mainly because AODV sends hello messages in order to sense neighbors. But both DSR and AODV suffer for large networks.

Packet delivery ratio becomes very small especially during high mobility for proactive distance vector protocol such as DSDV. This is because DSDV, which sends route information periodically, fails to catch up dynamic node movement. Even though DSDV avoids this problem by using triggered update, it faces another problem that is bandwidth consumption because it is using full dump at high mobility.

TORA, which is basically developed for ad-hoc networks, performs worst in the group of the protocols. Because it is a link state algorithm, it has a problem of delay, needs high processing power, and a very large routing overhead. In general, TORA is one of the largest and most complicated reactive protocols designed for dense network.

OLSR is a routing protocol which takes features from the traditional routing algorithm OSPF and optimizes its working operations so that it can best fit to the needs of MANET. In OLSR, normal flooding in link state algorithms during topology change is handled by using MPRs which greatly reduces routing overhead. MPRs also help OLSR to conserve energy as information is to be sent only to a node that is one hop away and in a bidirectional link. Since this optimized routing algorithm is not yet supported by NS-2, this

paper doesn't have any simulation results to discuss. My implementation also gives priority to a more efficient routing algorithm ZRP instead of OLSR.

Although it is called optimized, OLSR has a limitation as it is a proactive routing protocol. It implies that OLSR will fail when we have a more dynamic topology. Hence, a routing protocol can be efficient if it operates best under both proactive and reactive conditions. It should follow the characteristics of OLSR when the topology is static, and follow those of AODV/DSR when the topology becomes very dynamic. The ZRP routing protocol incorporates both these concepts – a hybrid routing protocol containing features from both proactive and reactive routing protocols.

7.3. Recommendation

As an output from this thesis, ZRP is to satisfy more or less all the requirements of MANET routing protocols especially during large network scenarios. ZRP can further be optimized if it uses the more efficient proactive routing protocol OLSR for its intrAzone routing and DSR/AODV for intErzone routing mechanism.

Finally this thesis recommends further enhancements to the current ad-hoc routing protocols; like considering

- Security
- Multicasting and
- Quality of Service.

One more additional observation is in the simulation software itself. The simulation software NS-2 has a scalability problem, when the size of the network becomes bigger and the simulation time becomes large. The log file becomes extremely large in the range of 100s of mega byte, and sometimes the simulation fails to finish. Because NS-2 is a discrete event simulator, this scalability problem of can be addressed by recording logs or trace values from widely spaced discrete events and also using a grid based approach.

8. Appendix I NS-2 Simulation Scripts

```
# dsdv.tcl
#
# simulation of wireless AODV MANET Routing Protocol
# with 50 nodes,
#
#
# Written by Etsegenet Awash
#
# ======================================================================
# Define options
# ======================================================================

set opt(chan)     Channel/WirelessChannel    ;# channel type
set opt(prop)     Propagation/TwoRayGround   ;# radio-propagation model
set opt(netif)    Phy/WirelessPhy            ;# network interface type
set opt(mac)      Mac/802_11                 ;# MAC type
set opt(ifq)      Queue/DropTail/PriQueue    ;# interface queue type
set opt(ll)       LL                         ;# link layer type
set opt(ant)      Antenna/OmniAntenna        ;# antenna model
set opt(ifqlen)   50                         ;# max packet in ifq
set opt(nn)       50                        ;# number of mobilenodes
set opt(adhocRouting)   DSDV                       ;# routing protocol

set opt(cp)             "/root/MANET/Traffic/cbr-50-"
                                ;# connection pattern file
set opt(sc)       "/root/MANET/Movement/scen-100-10"
                                ;# node movement file.

set opt(x)        500              ;# x coordinate of topology
set opt(y)        500               ;# y coordinate of topology
set opt(seed)     0.0               ;# seed for random number gen.
set opt(stop)     100                ;# time to stop simulation

#
======================================================================
===
# Main Program
#
======================================================================
===

#Change the value of sim to 0,10,20----90 (corresponding pause time)
#inorder to run different simulations.

      set sim 60

      # create simulator instance
      set ns_   [new Simulator]

      # trace files for ns and na,
      set tracefd  [open DSDV-out-$sim.tr w]
      set namtrace [open DSDV-out.nam w]

      # use new trace file format
```

```
$ns_ use-newtrace

$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

# Create topography object
set topo   [new Topography]

# define topology
$topo load_flatgrid $opt(x) $opt(y)

# create God
set god_   [create-god $opt(nn)]

# Define how to create a node
$ns_ node-config -adhocRouting $opt(adhocRouting) \
                 -llType $opt(ll) \
                -macType $opt(mac) \
               -ifqType $opt(ifq) \
               -ifqLen $opt(ifqlen) \
               -antType $opt(ant) \
               -propType $opt(prop) \
               -phyType $opt(netif) \
               -channelType $opt(chan) \
               -topoInstance $topo \
               -agentTrace ON \
               -routerTrace ON \
               -macTrace ON

# create the specifed number (nn) of nodes and attach them to the
# channel

for {set j 0} {$j < $opt(nn)} {incr j} {
      set node_($j) [ $ns_ node ]
      $node_($j) random-motion 0    ;# Disable random motion
}


# source connection-pattern and node-movement scripts

set filename $opt(cp)$sim-4

puts $filename

if { $filename == "" } {
      puts "*** NOTE: no connection pattern specified."
      set opt(cp) "none"
} else {
      puts "Loading connection pattern..."
      source $filename
}

#set filename $opt(sc)$sim

if { $opt(sc) == "" } {
      puts "*** NOTE: no scenario file specified."
      set opt(sc) "none"
```

```
} else {
      puts "Loading scenario file..."
      source $opt(sc)
      puts "Load complete..."
}

# Define initial node position in nam

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

      # 20 defines the node size in nam, must be adjusted
      # according to the scenario

      # The function must be called after mobility model is
      # defined

      $ns_ initial_node_pos $node_($i) 20
}

# Tell all nodes when the simulation ends
for {set i } {$i < $opt(nn) } {incr i} {
      $ns_ at $opt(stop).0001 "$node_($i) reset";
}

$ns_ at $opt(stop).0001 "puts \"NS EXITING ...\"; $ns_ halt"
$ns_ at $opt(stop).0002 "stop"

proc stop {} {
      global ns_ tracefd namtrace
      $ns_ flush-trace
      close $tracefd
      close $namtrace
}

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

```
# aodv.tcl
#
# simulation of wireless AODV MANET Routing Protocol
# with 100 nodes,
#
#
# Written by Etsegenet Awash
#
# ======================================================================
# Define options
# ======================================================================

set opt(chan)     Channel/WirelessChannel    ;# channel type
set opt(prop)     Propagation/TwoRayGround   ;# radio-propagation model
set opt(netif)    Phy/WirelessPhy            ;# network interface type
set opt(mac)      Mac/802_11                 ;# MAC type
set opt(ifq)      Queue/DropTail/PriQueue    ;# interface queue type
set opt(ll)       LL                         ;# link layer type
set opt(ant)      Antenna/OmniAntenna        ;# antenna model
set opt(ifqlen)   50                         ;# max packet in ifq
set opt(nn)       100                         ;# number of mobilenodes
set opt(adhocRouting)   AODV                       ;# routing protocol

set opt(cp)             "/root/MANET/Traffic/cbr-100-"
                                    ;# connection pattern file
set opt(sc)       "/root/MANET/Movement/scen-100-10"
                                    ;# node movement file.

set opt(x)        500               ;# x coordinate of topology
set opt(y)        500                ;# y coordinate of topology
set opt(seed)     0.0                ;# seed for random number gen.
set opt(stop)     50                 ;# time to stop simulation

#
=============================================================================
===
# Main Program
#
=============================================================================
===

#Change the value of sim to 10,20----90 (corresponding to traffic source)
inorder to run different simulations.

      set sim 60

      # create simulator instance
      set ns_   [new Simulator]

      # trace files for ns and na,
      set tracefd  [open AODV-out-$sim.tr w]
      set namtrace [open AODV-out.nam w]

      # use new trace file format
      $ns_ use-newtrace

      $ns_ trace-all $tracefd
```

```
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

# Create topography object
set topo   [new Topography]

# define topology
$topo load_flatgrid $opt(x) $opt(y)

# create God
set god_   [create-god $opt(nn)]

# Define how to create a node
$ns_ node-config -adhocRouting $opt(adhocRouting) \
                 -llType $opt(ll) \
                -macType $opt(mac) \
               -ifqType $opt(ifq) \
               -ifqLen $opt(ifqlen) \
               -antType $opt(ant) \
               -propType $opt(prop) \
               -phyType $opt(netif) \
               -channelType $opt(chan) \
               -topoInstance $topo \
               -agentTrace ON \
               -routerTrace ON \
               -macTrace ON

# create the specifed number (nn) of nodes and attach them to the
# channel

for {set j 0} {$j < $opt(nn)} {incr j} {
      set node_($j) [ $ns_ node ]
      $node_($j) random-motion 0    ;# Disable random motion
}


# source connection-pattern and node-movement scripts

set filename $opt(cp)$sim-4

puts $filename

if { $filename == "" } {
      puts "*** NOTE: no connection pattern specified."
      set opt(cp) "none"
} else {
      puts "Loading connection pattern..."
      source $filename
}

#set filename $opt(sc)$sim

if { $opt(sc) == "" } {
      puts "*** NOTE: no scenario file specified."
      set opt(sc) "none"
} else {
      puts "Loading scenario file..."
      source $opt(sc)
```

```
      puts "Load complete..."
}

# Define initial node position in nam

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

      # 20 defines the node size in nam, must be adjusted
      # according to the scenario

      # The function must be called after mobility model is
      # defined

      $ns_ initial_node_pos $node_($i) 20
}

# Tell all nodes when the simulation ends
for {set i } {$i < $opt(nn) } {incr i} {
      $ns_ at $opt(stop).0001 "$node_($i) reset";
}

$ns_ at $opt(stop).0001 "puts \"NS EXITING ...\"; $ns_ halt"
$ns_ at $opt(stop).0002 "stop"

proc stop {} {
      global ns_ tracefd namtrace
      $ns_ flush-trace
      close $tracefd
      close $namtrace
}

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

```
# dsr.tcl
#
# simulation of wireless DSR MANET Routing Protocol
# with 120 nodes,
#
#
# Written by Etsegenet Awash
#
# ======================================================================
# Define options
# ======================================================================

set opt(chan)     Channel/WirelessChannel    ;# channel type
set opt(prop)     Propagation/TwoRayGround   ;# radio-propagation model
set opt(netif)    Phy/WirelessPhy            ;# network interface type
set opt(mac)      Mac/802_11                 ;# MAC type
set opt(ifq)      CMUPriQueue                ;# interface queue type
set opt(ll)       LL                         ;# link layer type
set opt(ant)      Antenna/OmniAntenna        ;# antenna model
set opt(ifqlen)   50                         ;# max packet in ifq
set opt(nn)       120                        ;# number of mobilenodes
set opt(adhocRouting)   DSR                  ;# routing protocol

set opt(cp)       "/root/MANET/Traffic/cbr-120-36-4"
                                  ;# connection pattern file
set opt(sc)       "/root/MANET/Movement/scen-40-10"
                                  ;# node movement file.

set opt(x)        500              ;# x coordinate of topology
set opt(y)        500               ;# y coordinate of topology
set opt(seed)     0.0               ;# seed for random number gen.
set opt(stop)     50                ;# time to stop simulation



#
=============================================================================
===
# Main Program
#
=============================================================================
===


      #Change the value of sim to 0,10,20----200 (corresponding network #
size) inorder to run different simulations.
      set sim 10


      # create simulator instance
      set ns_   [new Simulator]

      # trace files for ns and na,
      set tracefd  [open DSR-out-$sim.tr w]
      set namtrace [open DSR-out.nam w]

      # use new trace file format
```

```
$ns_ use-newtrace

$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

# Create topography object
set topo   [new Topography]

# define topology
$topo load_flatgrid $opt(x) $opt(y)

# create God
set god_   [create-god $opt(nn)]

# Define how to create a node
$ns_ node-config -adhocRouting $opt(adhocRouting) \
                 -llType $opt(ll) \
                -macType $opt(mac) \
               -ifqType $opt(ifq) \
               -ifqLen $opt(ifqlen) \
               -antType $opt(ant) \
               -propType $opt(prop) \
               -phyType $opt(netif) \
               -channelType $opt(chan) \
               -topoInstance $topo \
               -agentTrace ON \
               -routerTrace ON \
               -macTrace OFF

# create the specifed number (nn) of nodes and attach them to the
# channel

for {set j 0} {$j < $opt(nn)} {incr j} {
      set node_($j) [ $ns_ node ]
      $node_($j) random-motion 0    ;# Disable random motion
}

# source connection-pattern and node-movement scripts

set filename $opt(cp)$sim-4

puts $filename

if { $filename == "" } {
      puts "*** NOTE: no connection pattern specified."
      set opt(cp) "none"
} else {
      puts "Loading connection pattern..."
      source $filename
}

#set filename $opt(sc)$sim

if { $opt(sc) == "" } {
      puts "*** NOTE: no scenario file specified."
      set opt(sc) "none"
} else {
```

```
      puts "Loading scenario file..."
      source $opt(sc)
      puts "Load complete..."
}

# Define initial node position in nam

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

      # 20 defines the node size in nam, must be adjusted
      # according to the scenario

      # The function must be called after mobility model is
      # defined

      $ns_ initial_node_pos $node_($i) 20
}



# Tell all nodes when the simulation ends
for {set i } {$i < $opt(nn) } {incr i} {
      $ns_ at $opt(stop).0001 "$node_($i) reset";
}



$ns_ at $opt(stop).0001 "puts \"NS EXITING ...\"; $ns_ halt"
$ns_ at $opt(stop).0002 "stop"


proc stop {} {
      global ns_ tracefd namtrace
      $ns_ flush-trace
      close $tracefd
      close $namtrace
}


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

```
# tora.tcl
#
# simulation of wireless TORA MANET Routing Protocol
# with 120 nodes,
#
#
# Written by Etsegenet Awash
#
# ======================================================================
# Define options
# ======================================================================

set opt(chan)     Channel/WirelessChannel    ;# channel type
set opt(prop)     Propagation/TwoRayGround   ;# radio-propagation model
set opt(netif)    Phy/WirelessPhy            ;# network interface type
set opt(mac)      Mac/802_11                 ;# MAC type
set opt(ifq)      Queue/DropTail/PriQueue    ;# interface queue type
set opt(ll)       LL                         ;# link layer type
set opt(ant)      Antenna/OmniAntenna        ;# antenna model
set opt(ifqlen)   50                         ;# max packet in ifq
set opt(nn)       120                        ;# number of mobilenodes
set opt(adhocRouting)   TORA                 ;# routing protocol

set opt(cp)        "/root/MANET/Traffic/cbr-120-36-4"
                                  ;# connection pattern file
set opt(sc)       "/root/MANET/Movement/scen-120-10"
                                  ;# node movement file.

set opt(x)        500              ;# x coordinate of topology
set opt(y)        500               ;# y coordinate of topology
set opt(seed)     0.0               ;# seed for random number gen.
set opt(stop)     50                ;# time to stop simulation


#
# ======================================================================
===
# Main Program
#
# ======================================================================
===


#Change the value of sim to 40, 80, 120 --- 200 (corresponding network #
size) inorder to run different simulations.
      set sim 120


      # create simulator instance
      set ns_   [new Simulator]

      # trace files for ns and na,
      set tracefd  [open TORA-out-$sim.tr w]
      set namtrace [open TORA-out.nam w]

      # use new trace file format
      $ns_ use-newtrace
```

```
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

# Create topography object
set topo   [new Topography]

# define topology
$topo load_flatgrid $opt(x) $opt(y)

# create God
set god_   [create-god $opt(nn)]

# Define how to create a node
$ns_ node-config -adhocRouting $opt(adhocRouting) \
                 -llType $opt(ll) \
                -macType $opt(mac) \
               -ifqType $opt(ifq) \
               -ifqLen $opt(ifqlen) \
               -antType $opt(ant) \
               -propType $opt(prop) \
               -phyType $opt(netif) \
               -channelType $opt(chan) \
               -topoInstance $topo \
               -agentTrace ON \
               -routerTrace ON \
               -macTrace OFF

# create the specifed number (nn) of nodes and attach them to the
# channel

for {set j 0} {$j < $opt(nn)} {incr j} {
      set node_($j) [ $ns_ node ]
      $node_($j) random-motion 0    ;# Disable random motion
}


# source connection-pattern and node-movement scripts

#set filename $opt(cp)$sim-4


if { $opt(cp) == "" } {
      puts "*** NOTE: no connection pattern specified."
      set opt(cp) "none"
} else {
      puts "Loading connection pattern..."
      source $opt(cp)
}

#set filename $opt(sc)$sim

if { $opt(sc) == "" } {
      puts "*** NOTE: no scenario file specified."
      set opt(sc) "none"
} else {
      puts "Loading scenario file..."
```

```
        source $opt(sc)
        puts "Load complete..."
}


# Define initial node position in nam

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

        # 20 defines the node size in nam, must be adjusted
        # according to the scenario

        # The function must be called after mobility model is
        # defined

        $ns_ initial_node_pos $node_($i) 20
}


# Tell all nodes when the simulation ends
for {set i } {$i < $opt(nn) } {incr i} {
        $ns_ at $opt(stop).0001 "$node_($i) reset";
}



$ns_ at $opt(stop).0001 "puts \"NS EXITING ...\"; $ns_ halt"
$ns_ at $opt(stop).0002 "stop"


proc stop {} {
        global ns_ tracefd namtrace
        $ns_ flush-trace
        close $tracefd
        close $namtrace
}


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

```
# zrp.tcl
```

```
#
# simulation of wireless ZRP MANET Routing Protocol
# with 100 nodes,
#
#
# Written by Etsegenet Awash
#
# ======================================================================
# Define options
# ======================================================================

set opt(chan)     Channel/WirelessChannel    ;# channel type
set opt(prop)     Propagation/TwoRayGround   ;# radio-propagation model
set opt(netif)    Phy/WirelessPhy            ;# network interface type
set opt(mac)      Mac/802_11                 ;# MAC type
set opt(ifq)      Queue/DropTail/PriQueue    ;# interface queue type
set opt(ll)       LL                         ;# link layer type
set opt(ant)      Antenna/OmniAntenna        ;# antenna model
set opt(ifqlen)   50                         ;# max packet in ifq
set opt(nn)       100                        ;# number of mobilenodes
set opt(adhocRouting)   ZRP                  ;# routing protocol

set opt(cp)       "/root/MANET/Traffic/cbr-100-20-8"
                                            ;# connection pattern file
set opt(sc)       "/root/MANET/Movement/scen-100-10"
                                            ;# node movement file.

set opt(x)      1000                        ;# x coordinate of topology
set opt(y)      [expr ($opt(nn)+1)*250]     ;# y coordinate of topology
#set opt(y)     500                         ;# y coordinate of topology
set opt(seed)   0.0                         ;# seed for random number gen.
set opt(stop)   200                         ;# time to stop simulation


#
======================================================================
===
# Main Program
#
======================================================================
===

# Change the value of sim to 1, 2, 3 --- 8 (corresponding zone radius)
# inorder to run different simulations.

      set sim 1

      # create simulator instance
      set ns_   [new Simulator]

      # trace files for ns and na,
      set tracefd  [open ZRP-out-$sim.tr w]
      set namtrace [open ZRP-out.nam w]
      # use new trace file format
      $ns_ use-newtrace

      $ns_ trace-all $tracefd
```

```
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

# Create topography object
set topo   [new Topography]

# define topology
$topo load_flatgrid $opt(x) $opt(y)

# create God
set god_   [create-god $opt(nn)]

# Define how to create a node
$ns_ node-config -adhocRouting $opt(adhocRouting) \
                 -llType $opt(ll) \
                -macType $opt(mac) \
               -ifqType $opt(ifq) \
               -ifqLen $opt(ifqlen) \
               -antType $opt(ant) \
               -propType $opt(prop) \
               -phyType $opt(netif) \
               -channelType $opt(chan) \
               -topoInstance $topo \
               -agentTrace ON \
               -routerTrace ON \
               -macTrace ON \
               -movementTrace OFF \

# create the specifed number of nodes and attach them to the
# channel

for {set j 0} {$j < $opt(nn)} {incr j} {
      set node_($j) [ $ns_ node ]
      $node_($j) random-motion 0    ;# Disable random motion
}

# source connection-pattern and node-movement scripts
if { $opt(cp) == "" } {
      puts "*** NOTE: no connection pattern specified."
      set opt(cp) "none"
} else {
      puts "Loading connection pattern..."
      source $opt(cp)
}

#set filename $opt(sc)$sim
if { $opt(sc) == "" } {
      puts "*** NOTE: no scenario file specified."
      set opt(sc) "none"
} else {
      puts "Loading scenario file..."
      source $opt(sc)
      puts "Load complete..."
}
# Define initial node position in nam

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

```
      # 20 defines the node size in nam, must be adjusted
      # according to the scenario

      # The function must be called after mobility model is
      # defined

      $ns_ initial_node_pos $node_($i) 20
}

#Mac/802_11 set dataRate_ 1Mb

# Define zone radius (radius = 3.0)
for {set k 0} {$k < $opt(nn)} {incr k} {

      set r_($k) [$node_($k) set ragent_]
      $ns_ at 0.0 "$r_($k) radius $sim"

}


# Tell all nodes when the simulation ends
for {set i } {$i < $opt(nn) } {incr i} {
      $ns_ at $opt(stop).0001 "$node_($i) reset";
}


$ns_ at $opt(stop).0001 "puts \"NS EXITING ...\"; $ns_ halt"
$ns_ at $opt(stop).0002 "stop"


proc stop {} {
      global ns_ tracefd namtrace
      $ns_ flush-trace

      close $tracefd
      close $namtrace
}


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

9. Bibliography

[1] D.Johnson et al., Dynamic Source Routing for mobile Ad-hoc Networks, IETF MANET Draft, April2003

[2] Charles E.Perkins, Destination Sequenced Distance Vector Protocol, Ad-hoc Networking

[3] Charles E.Perkins, Elzizabeth M.Belding-Royer, Samir R.Das, Ad-hoc On-demand Distance Vector Routing, IETF MANET Draft, February2003. http://www.ietf.org/rfc/rfc3561.txt cited 01.03.2004

[4] Vincent D.Park, M.scott Corson. A highly Adaptive Distributed Routing Algorithm for Mobile wireless Networks. http://www.cs.odu.edu/skovvuri/tora.pdf cited 10.04.2004

[5] T.Clausen and P.Jacquent Optimized Link State Routing Protocol (OLSR). RFC3626, IETF Network Work Group, October2003

[6] Zj.Haas and M.R Pearlman, The Zone Routing Protocol (ZRP), for Ad-hoc Networks. IETF Internet Draft, draft-ietf-manet-zone-zrp-04.txt,July 2002

[7] Tsu –Wei Chen and Mario Gerla, Global State Routing : A new routing scheme for Ad-hoc wireless networks. http://www.ics.uci.edu/~atm/adhoc/paper

[8] Mingliang Jiang, Jinyang Li, Y.C. Tay, Cluster Based Routing Protocol, August1999, IETF, Draft

[9] A.Iwata, C-C Chiang, G.Pei, M.Gerla and T.-W-chen, Scalable Routing Strategies for Ad-hoc wireless Network, IEEE Journal on selected Area in communication, Special Issues on Ad-hoc Network, August1999.pp 1368-79. http://www.cs.ucla.edu/NRL/wireless/paper/Jsac99.ps.gz

[10] S.Corson, J.Macker. MANET: Routing Protocol Performance Issues and Evaluation Considerations RFC 2501, IETF Network Working Group January1999, http://www.ietf.org/rfc/rfc2501.txt cited 15.02.2004

[11] M.Steenstrup, editor. Routing in Communication Networks, Perntice Hall,Inc,Engewood Cliffs,NJ,1995.

[12] A.B. Mc Donald and T.F.Znati: A mobility-based framework for adaptive clustering in wireless Ad-hoc networks. IEEE journal on Selected Areas in Communications, august1999

[13] Xiauyan Hong. Kaixin Xu and Mario Gerla, Scalable Routing Protocols for mobile Ad-hoc Networks, Computer Science Deparetment, University of California, Los Angeles, August2002.

[14] A.Lauti P.MÜhlethaler, A.Najid and E.Plakoo, Simultion Results of the OLSR Routing Protocols for wireless Networks, 1st Mediterranean Ad-hoc Networks Workshop (Med-Hoc-Net). Sardegna, Itally 2002

[15] V.Park, S.Corson, Temporally-Ordered Routing Algorithm (TORA) Version I functional specification. Internet Draft, draft-ietf-manet-tora spec01.txt, August7,1998

[16] Jan. Schaumann. Analysis of The Zone Routing Protocol. http://www.netmeister.org/misc/zrp/zrp.pdf , December 8,2002.

[17] Z.J. Haas and M.R. Pearlman. The Performance of Query Control Schemes for the Zone Routing Protocol. ACM/IEEE Transactions on Networking Vol.9,no.4, August2001.

[18] M. JIANG, J. LI, Y. C. TAY *Cluster Based Routing Protocol (CBRP) Functional Specification* Internet Draft, draft-ietf-manet-cbrp.txt, *work in progress*, June 1999.

[19] The network simulator NS-2, NS-2 Manual, http://www.isi.edu./nsnam/ns

[20] Josh Borch, David A.Maltz, David B.Johnson, Yih-Chun Hu and Jorjeta Jetcheva, A performance Comparison of Multi-hop Ad-hoc Networking Protocols, Computer Science Department, Carnegie Mellon University Pittburgh, PA 15213 http://www.monarch.cs.cmu.edu

[21] J.Moy OSPF version2 RFC 1247, july1991