MULTICASTING
IN
AD-HOC NETWORKS

By

Shah Ankit K.

(05MCE014)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY OF SCIENCE & TECHNOLOGY

Ahmedabad 382481

May 2007

This is to certify that Dissertation entitled

Multicasting
In
Ad-Hoc Networks

Submitted by

Ankit K. Shah

has been accepted toward fulfillment of the requirement
for the degree of
Master of Technology in Computer Science & Engineering

Prof. Sharada Valiveti	Prof. D. J. Patel
Professor In Charge	Head of The Department

Prof. A. B. Patel
Director, Institute of Technology

CERTIFICATE

This is to certify that the Major Project entitled "Multicasting In Ad-Hoc Networks" submitted by Mr. Ankit K Shah (05MCE014), towards the partial fulfillment of the requirements for the degree of Master of Technology in Computer Science & Engineering of Nirma University of Science and Technology, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Prof. Sharada Valiveti
Guide
Asst. Professor,
Department of Computer Engineering,
Institute of Technology,
Nirma University,
Ahmedabad

Date :-

ACKNOWLEDGEMENT

I would like to express my humble gratitude towards the mentor and guide of my project dissertation – Dr. S. N. Pradhan (Senior Professor and M.Tech. Co-ordinator) who has always inspired all of us to put in the maximum of our efforts into the dissertation work. He has always been helpful to provide the necessary facilities and also to solve our problems, which may have occurred during our ongoing project work.

My sincere thanks to my guide – Asst. Prof. Sharada Valiveti to continuously guide me in my project area and to provide me with all the necessary resources and material which have been extremely useful in my dissertation. I am highly obliged for her constant overwhelming support.

I would like to thank Prof. A. B. Patel (Director, NIT) and Prof. D. J. Patel (H.O.D., CSE Dept.) for supervising the entire dissertation program and organizing meetings in order to receive feedback from students as well as the staff-in-charge regarding the problems faced in the program and their efforts to solve them to their best.

Last, but not at all the least, I thank all the people, either directly or indirectly related, involved or concerned in helping me with the project dissertation work through their moral support; be it my family, my classmates or my friends.

— ANKIT SHAH —
(05MCE014)

ABSTRACT

Multicast communication has been used for a wide range of applications. Multicast is a more efficient method of supporting group communication than unicast, as it allows transmission and routing of packets to multiple destination using fewer network resources. Within the wired network, well established wired protocols exist to offer an efficient multicast service. As node becomes increasingly mobile, protocols need to evolve to provide efficient service in the new environment.

MAODV protocol is based on the AODV unicast routing protocol, which is a tree based multicast routing protocol and uses a group leader to maintain a multicast group. Mainly two approaches are studied and analyzed, one is to make the group leader node static and then measure the parameters like Packet Delivery Ratio and Latency. And the other, in which, new packet format of RI (Receiver Identification) is designed, which is sent periodically by sender to check that the receiver node is in the range of the sender node; if it is in the range of the sender node than sender can directly send packets to receiver rather than sending it through intermediate nodes. Petri Net model approach has been used to show that MAODV is a complete and safe protocol.

Network Simulator 2 tool is used for coding and simulation purpose of the modified MAODV protocol. The newly implemented MAODV is being used in this project to analyze its performance in different scenarios and are compared with the MAODV protocol without changes.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AODV	Ad-Hoc on demand routing protocol
CBR	Constant Bit Rate
GRPH	Group Hello
MAC	Medium Access Control
MACT	Multicast Activation
MANET	Mobile Ad-Hoc Networks
MAODV	Multicast ad-hoc on demand distance vector routing protocol
NS2	Network Simulator
PDR	Packet Delivery Ratio
RREP	Route Reply
RREQ	Route Request
RI	Receiver Identification
TTL	Time-to-Live

1 INTRODUCTION

1.1 MOBILE AD-HOC NETWORKS

1.1.1 General

A wireless ad-hoc network is a collection of mobile nodes with no pre established infrastructure, forming a temporary network. Each of the nodes has a wireless interface and communicates with each other with either radio or infrared. Laptop computers and personal digital assistants that communicate directly with each other are some examples of nodes in ad-hoc network [1]. Nodes in the ad-hoc networks are often mobile, but can also consist of mobile nodes, such as access point to the Internet. Semi nodes can be used to deploy relay points in areas where relay points might be needed temporarily.

An ad-hoc network uses no centralized administration. This is to be sure that the network won't collapse just because one of the mobile node moves out of transmitter range of the others. Nodes should be able to enter/leave the network as they wish. Because of the limited transmitter range of the nodes, multiple hops are needed to forward packets to other nodes. Thus every node acts both as a host and as a router.

Ad-hoc networks are also capable of handling topology changes and malfunctions in nodes. It is fixed through network reconfiguration. For instance, if a node leaves the network and causes link breakages, affected nodes can easily request new routes and the problem will be solved. This will slightly increase the delay but the network will still be operational [1].

Figure 1.1 shows a simple ad-hoc network with three nodes. The outermost nodes are within transmitter range of each other. However the middle nodes can be used to forward packets between the outermost nodes. The middle node is acting as a router and the three nodes have formed an ad-hoc network.

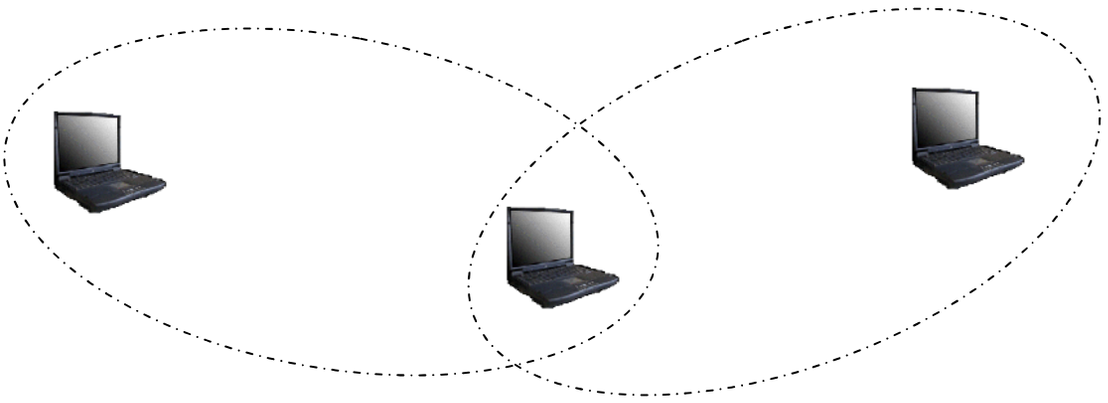


Figure 1.1 Example of simple ad-hoc networks with three participating nodes

1.1.2 Usage

Ad hoc network due to their fast and economically less demanding deployment find application in many areas [1]. Some of them are military applications, emergency operations, wireless mesh network, collaborative and distributed computing, etc.

Emergency Operations

Ad hoc networks are very useful in areas of search and rescue operations. The major factor that serves the purpose is minimal configuration and overhead, independent of fixed or centralized structure and the freedom of flexibility of movement. In such areas where the current infrastructure based networks are destroyed, ad hoc networks are very useful.

Military Applications:

Ad hoc networks due to their quick deployment are very useful in establishing communication among soldiers for tactical operations. Setting of fixed network in enemy territories may not be possible. For this ad hoc network provides the solution.

Collaborative and Distributed Computing:

Another domain where ad hoc network finds its place is in the area of collaborative computing. The requirement of temporary communication infrastructure, for quick communication with minimal configuration among a group of people in a conference or a gathering necessitates the formation of ad hoc wireless network. For example consider a group of people who wanted to share a file of research or any presentation in conference or lecture distributing notes on the fly. In such environment ad hoc network with reliable multicast protocol can serve the purpose**.**

1.2 DESIGN CHALLENGES FOR MULTICASTING PROTOCOL

The peculiarities of broadcast radio channels and the mobility of the nodes are the major reasons why wireless networks pose research challenges that are not present in a wired internet environment. One example is the vulnerability of wireless channels to a problem known as hidden terminals (Figure 1.2), where a central node A can reach other two nodes B and C every time it transmits data through its radio interface, but node B cannot reach node C directly with its own and vice-versa.

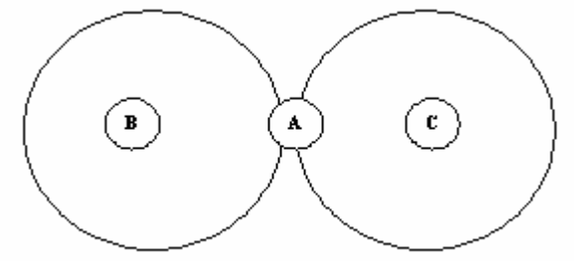
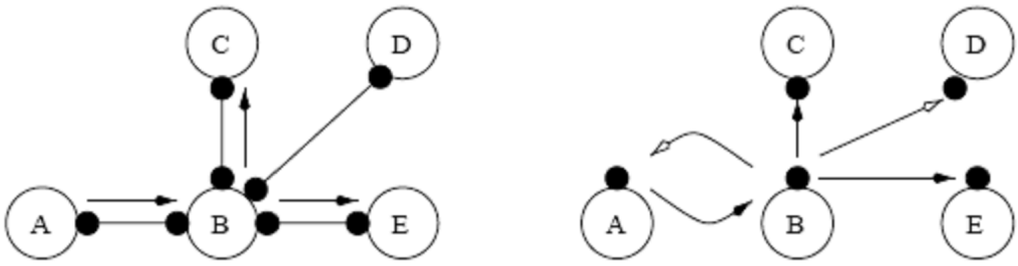


Figure 1.2 Hidden Terminal Problem

This problem creates a technical challenge for the design of multicast solutions targeting ad-hoc networks. In wired environments, data packet forwarding is done by a router in a per-interface basis. The packet is taken from the incoming interface and is copied to all other interfaces that lead to other members of the multicast groups. To avoid multicast routing loops, the incoming packet is never copied to the incoming interface. However, in ad-hoc networks, due to hidden-

terminals, very frequently the incoming packet must be copied to the incoming interface, and other precautions have to be taken to avoid looping.



(a) (b)

Figure 1.3: Multicast data packet forwarding

(a) Wired point to point network

(b) A wireless network

Figure 1.3 illustrates the problem. A small network with point-to-point links is represented in (Figure 1.3a), where node A sends a data packet to node B, which in turn forwards the packet only to multicast member nodes, i.e., nodes C and E. But as it can be seen in (Figure 1.3b), accomplishing the same task in a broadcast wireless network implies in overhead traffic, not only for node D, which is not a multicast member but also to the sender node A, who will get a copy of its packet as node B rebroadcasts it. Therefore, multicast group members in a wireless multihop network like node A have to use a mechanism to detect packets it has already seen to avoid duplication.

This thesis explores the provision of multicasting over wireless mobile networks. One key aspect addressed is the routing of multicast data from one or more senders to all participants of a multicast group in such a way that path exists between sources and receivers even when some or all of them move. The other aspect investigated is the reliable dissemination of information from a source to receivers in a mobile wireless network such that all data generated.

Limited bandwidth availability, the mobility of nodes with limited energy resources, the hidden terminal problem, and limited security make the design of the multicast protocol for ad-hoc networks a challenging one. The following are the some issues that should be taken care of while designing a multicasting protocol [1].

Robustness: Due to mobility of nodes, link failures are quite common in ad-hoc networks. Thus, data packets sent by the source may be dropped; which results in lower packet delivery ratio. Hence a multicast protocol should be robust enough to sustain the mobility of the nodes and achieve a high packet delivery ratio.

Efficiency: In ad-hoc network environment, where the bandwidth is scare, the efficiency of the multicast protocol is very important. Multicast efficiency is defined as the ratio of the total number of data packet received to the total no of data packet transmitted.

Control overhead: In order to keep track of members in a multicast group, the exchange of control packet is required. So, the designed multicast protocol ensures that total number of control packet transmitted for maintaining a multicast group is kept to a minimum.

Quality of service: The main parameters which are to be taken into consideration for providing the required QoS are throughput, delay, jitter, and reliability.

Resource Management: Ad hoc network consists of a group of mobile nodes, with each node having limited battery power and memory. An ad hoc multicast routing protocol should use minimum power by reducing the number of packet transmissions. To reduce memory usage, it should use minimum state information.

1.3 PROBLEM STATEMENT

Wireless networks are a convenient way of commutating because computer and infrastructure devices are not cabled to each other. These networks constitute an environment where the user is free to move beyond the confines of an office. One can take his/her computer and notes to the next door, next building or maybe the next city and still be able to exchange information with colleagues.

The very characteristic that makes wireless networks convenient also makes them a very challenging technology to implement. The fact that nodes may move around the network makes it necessary for network hardware and almost all software's to be designed from the ground up with mobility in mind. Many facts taken for granted in wired internets do not apply in the wireless scenario.

The core of our approach and contributions on multicasting over wireless networks consists of enabling the dissemination of multicast data packets over connected trees rather than meshes, which had been the norm in all prior approaches to multicasting in wireless and wired networks. Multicast tree are an important implementation for multicasting over wireless networks because tree based protocols are more efficient than mesh based protocols; furthermore, in tree based protocols source as well as receiver can initiate multicast session to transfer data packets.

1.4 ASSUMPTIONS

Here we have used Network Simulator for the simulation. So we have to follow simulator assumptions. The ranges for the mobile nodes are fixed i.e. that is 250 meter using omni directional antenna. All nodes in the simulation are moving randomly and IP address assigned to them during simulation are not real ones. At the start of the simulation each receiving node first joins the multicast tree. Nodes are moving in the fixed predefined area and when they reach the boundary, they just bounce back in to the area and start moving.

1.5 OUTLINE OF THESIS

Chapter 2 gives the overview of multicasting in ad-hoc networks and describes the types of protocols for multicasting in ad-hoc networks and also describes the architecture.

Chapter 3 describes the concept of multicast routing over trees and presents the main features of the multicast ad-hoc on demand distance vector routing (MAODV). Chapter 4 addresses the correctness of MAODV, which implies that no

deadlocks occur, that trees are built within a finite time, and that no data packets loop, using a Petri net approach.

Chapter 5 gives introduction to Network Simulator tool, which is been used in this thesis. Chapter 6 covers the implementation detail of the MAODV protocol in Network Simulator. Chapter 7 shows the results of simulation experiments comparing the performance of MAODV against original MAODV protocol. Chapter 8 presents a summary of our contributions and points out future research directions.

2 LITERATURE SURVEY

2.1 GENERAL

Multipoint communication has been around for a relatively long time. According to the survey by Diot et al. [2], research around the idea of one or more nodes sending data to many receivers simultaneously, has been available at least since the early 80's. What was known as multipoint communication became more widely called multicasting only after Deering established the first internet request-for-comments on the field [3].

The usage of the MBONE has declined for the past few years. However, on the other end of the spectrum, a few years back multicast service was made available by major telecommunication companies and national Internet service providers, finally bringing the technology closer to the corporate networking mainstream.

Most of the research and development efforts so far consider physically connected networks, where computers are brought together by the use of cables and wires. Because of the wired connections among them, the location of nodes is static and the only dynamics in the networks come from node and link failures.

Wireless networks are a very attractive way of computing because computer and infrastructure devices are not cabled to each other. These networks constitute an environment where the user is free to move beyond the confines of an office. The very characteristic that makes wireless networks convenient also makes them a very challenging technology to implement. The fact that nodes may move around the network makes it necessary for network hardware and almost all software be designed from the ground up with mobility in mind. Many facts taken for granted in wired internets do not apply in the wireless scenario [4].

Many routing protocols have been proposed [2, 4, 6], but few comparisons between the different protocols have been made. There exist some other simulation results that have been done on individual protocols. These simulations

have however not used the same metrics and are therefore not comparable with each other.

Within the wired network, well established routing protocols exist to offer efficient multicasting service. Adopting wired multicast protocols to a MANET, which completely lack any infrastructure, appears less promising. The protocols, having been designed for fixed networks, may fail to keep up with node movements and frequent topology changes due to host mobility, and the protocol overheads may increase substantially. Rather, new protocols that operate in an on-demand manner are being proposed and investigated. Existing studies show that tree-based on-demand protocols are not necessarily the best choice for all applications [4]. In harsh environment, where the network topology changes very frequently, mesh-based protocols seem to outperform tree-based protocols due to the availability of alternate paths, which allow multicast datagrams to be delivered to all or most multicast receivers even if some links fail. Between tree-based and mesh-based approaches, we can find hybrid protocols suitable for medium mobility networks taking advantage of both a tree and a mesh structure [5, 6].

2.2 OPERATIONS OF THE MULTICAST ROUTING PROTOCOLS

Based on the type of operation, multicast protocols for ad hoc wireless networks are broadly classified into two types: source-initiated protocols and receiver-initiated protocols [1].

2.2.1 Source Initiated Protocols

Figure 2.1 shows the event as they occur in a source-initiated protocol that uses a soft state approach. In a soft state maintenance approach, the multicast tree or mesh is periodically updated by means of control packets. In such protocols, the sources of the multicast group periodically flood a Join Request packet through the network. This JionReq is propagated by other nodes in the network, and it eventually reaches all the receivers in the group. Receivers of the multicast groups express their wish to receive packets from the group by responding with a JoinReply packet, which is propagated along the reverse path of that followed by the JoinReq packet. This JoinReply packet establishes forwarding states in the

intermediate nodes, and finally reaches the source. Thus, this is a two-pass protocol for establishing the tree (or mesh). There is no explicit procedure for route repair. In soft state protocols, the source periodically (once every refresh period) initiates the above procedure.

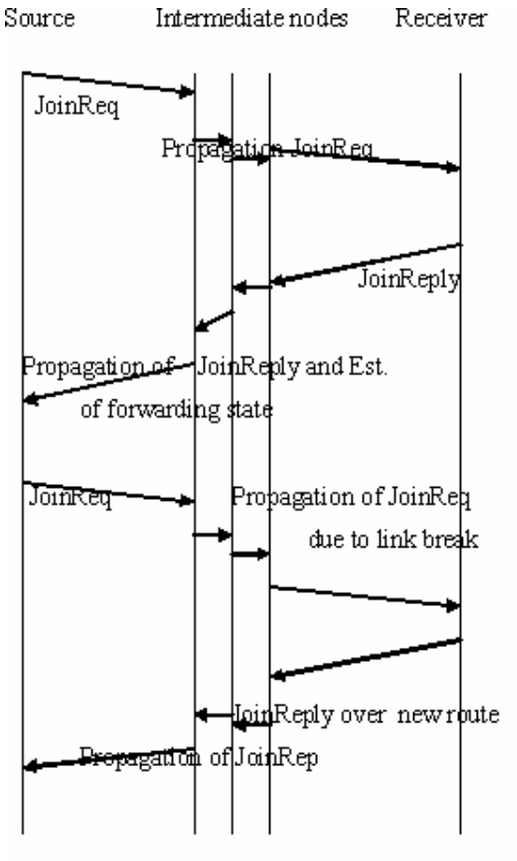


Figure 2.1 Source initiated protocol (Soft state approach)

Figure 2.2 shows the operation of the hard state source-initiated protocol. It is similar to that of a soft state source-initiated protocol, except that there is an explicit route repair procedure that is initiated when link break is detected. The route repair procedure shown in Figure 2.2 is a simple solution. The upstream node, which detect that one of its downstream nodes has moved away, initiates a tree construction procedure. Different protocol adapts different strategies for route repair.

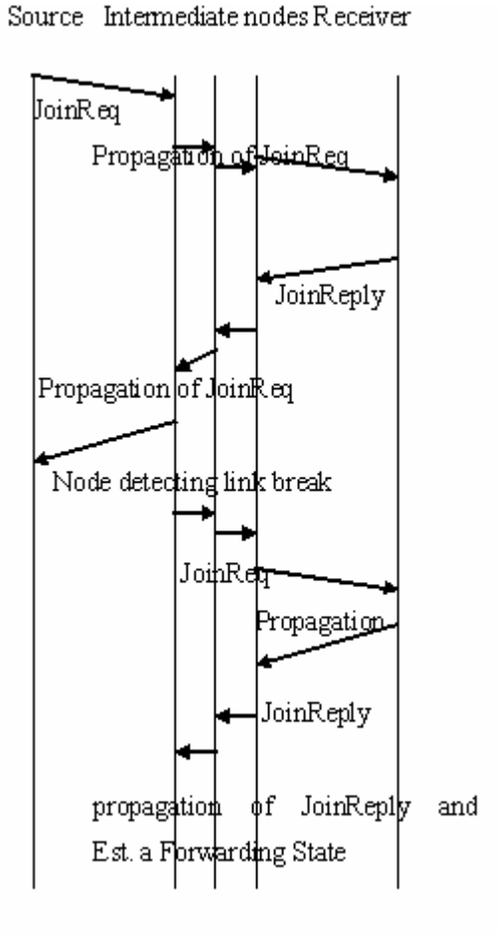


Figure 2.2 Source initiated protocol (Hard state approach)

Some protocols choose to have the downstream node search for its former parent by means of limited flooding, while others impose this responsibility on the upstream nodes.

2.2.2 Receiver Initiated Protocol

In the receiver initiated multicasting protocol, receiver uses flooding to search the paths to the source of the multicast groups to which it belongs. The soft state variant is illustrated in Figure 3. The tree construction is a three phase process. First, the receiver floods a JoinReq packet, which is propagated by the other nodes. Usually the source of the multicast groups and/or nodes which are already part of the multicast tree, are allowed to respond to JoinReq packet with a JoinReply packet, indicating that they would be able to send the data for the multicasting group.

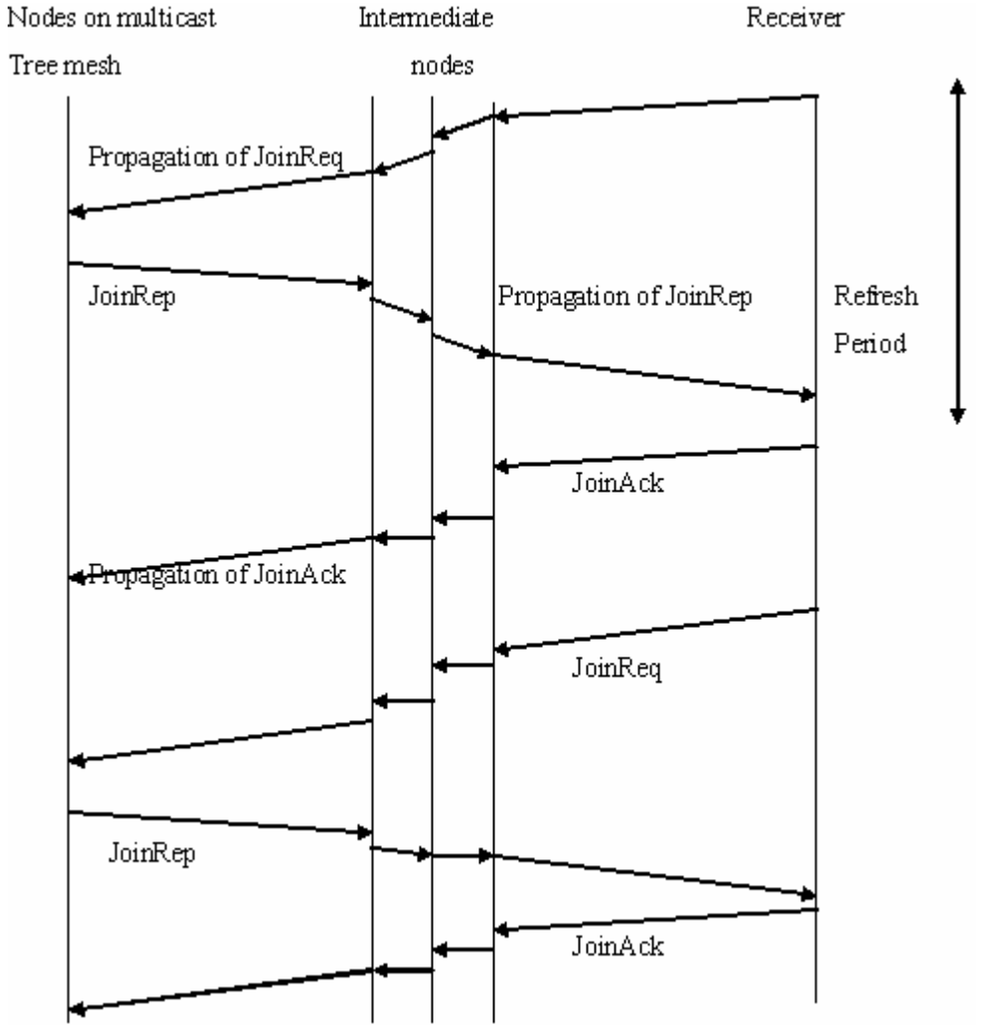


Figure 2.3 Receiver initiated protocol (Soft state approach)

The receiver chooses the JoinReply with the smallest hop count and JoinAck packet along the reverse path. Route maintenance is accomplished by the periodic initiation of this procedure by the receiver.

Figure 2.3 shows the hard state variant. The initial tree-joining phase is the same that in corresponding soft state protocol. The route repair procedure comes in to play when a link break is detected: the responsibility to restore the multicast topology can be described to wither the downstream or the upstream node. In Figure 2.3, the downstream node searches for the route to the multicast tree through a procedure similar to the initial topology construction procedure

2.3 AN ARCHITECTURAL REFERENCE MODEL FOR MULTICAST ROUTING PROTOCOL

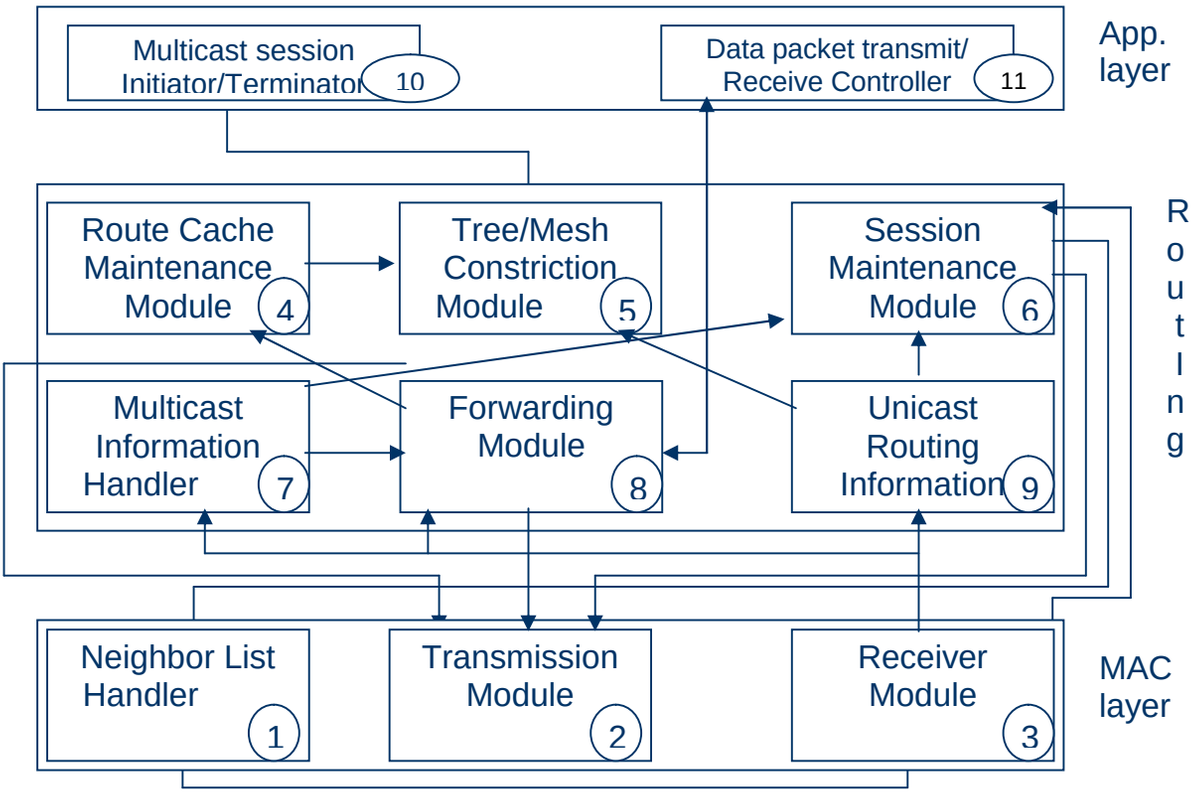


Figure 2.4 Architecture reference model

Figure 2.4 shows the architecture reference model multicast routing protocol [1].

2.3.1. Medium Access Control (MAC) Layer

The important services provided by this layer to the ones above are transmission and reception of packets. Apart from these functions, three other important functions are performed by this layer that is particularly important for wireless multicast:

A. Transmission module:

This module also includes the arbitration module, which schedules transmission on the channel. The exact nature of this scheduling depends on the MAC protocol. In general, the MAC protocol might maintain multicast state information based on the past transmission

observed on the channel, and the scheduling is dependent on the state information.

B. Receiver Module:

Same as the transmission module

C. Neighbor List Handler

This module informs the higher layer whether a particular node is a neighbor or not. In maintains a list of the neighboring nodes. This functionality can be implemented by the means of the beacon or by overhearing all the packets on the channel.

2.3.2 Routing Layer

This layer is responsible for forming and maintaining the unicast session/multicast group. For this purpose, it uses a set of tables, timers, and route caches. The important service it provides to the application layer are the functions to join/leave multicast group and to transmit/receive multicast packets.

A. Unicast Routing Information Handler

This serves to discover unicast routes.

B. Multicast information handler

This maintains all the pertinent information related to the state of the current node with respect to the multicast group of which it is a part, in the form of table. This state might include a list of its downstream nodes, the address of its upstream nodes, sequence number information, etc. This table might be maintained per group or per source per group.

C. Forwarding module

This uses the information provided by the multicast information handler to decide whether a received multicast packet should be broadcast, or be forwarded to a neighboring node, or be sent to the application layer

D. Tree/mesh construction module

This module is used to construct the multicast topology. It can use information provided by the unicast routing information handler for this purpose; for example this module might initiate flooding, which is being requested to join a group by a application layer. Also when the application layer process sends the session termination message to this module, this module transmits the appropriate message to the network for terminating the participation of the current node in the group.

E. Session maintenance module

This module initiates a route repair on being informed of a link break by a lower layer. It might use information from the unicast and multicast tables to perform a search for a node in order to restore the multicast topology.

F. Route cache maintenance module

The purpose of this module is to glean information from routing packets overhead on the channel for possible use later. Such information might be the address of nodes which have requested for a route to a multicast group source, etc. The route cache is updated as newer information obtained from the more recent packet heard on the channel. This module is usually optional in most multicast protocols. It increases efficiency by reducing the control overhead.

2.3.3 Application Layer

This layer utilizes the services of the routing layer to satisfy the multicast requirements of applications. It primarily consists of two modules:

a) Data packet transmit/receive module
b) Multicast session initiator/terminator

All the above modules and interaction between them is shown in Figure 2.4. The interactions between these modules can be understood by considering some action that take place during the life time of the multicast session:

1. **Joining a group**

   Module 10, which exists in the application layer, makes a request to join a group to module 5 present in the routing layer, which can cache information from the module 4 and unicast route information from the module 9. It then initiates flooding of JoinReq packets by using module 2 of the MAC layer. These JoinReq are passed by module 3 of other nodes to their forwarding module, which updates the multicast tables and propagate this message. During the reply phase, the forwarding states in the multicast tables of intermediate nodes are established.

2. **Data packet propagation**

   Data packets are handled by module 11 in the application layer, which passes them on to module 8, which makes the decision on whether to broadcast packets after consulting module 7. a similar occurs in the all nodes belonging to the multicast topology until eventually the data packets are sent by the forwarding module of the receivers to the application layer.

3. **Route repair**

   Route repair is handled by module 6 on being informed by the module 1 of link breaks. It uses the unicast and multicast routing tables to graft the node back into the multicast topology.

2.4 CLASSIFICATION OF A MULTICAST ROUTING PROTOCOLS

Multicast routing protocols for ad hoc wireless networks can be broadly classified in to two types: application-independent/generic multicast protocols and application-dependent multicast protocols [1]. Application-independent multicast protocols can be classified along three different dimensions.

2.4.1 Based on topology

Current protocols used for ad hoc multicast routing protocols can be classified in to two types based on the multicast topology: **tree based** and **mesh based**. In tree based protocols, there exists a single path between a source and the receiver, where in mesh based protocols; there may be more than one path between source and receiver. Tree based protocol are more efficient compared to mesh based protocols, but mesh based protocol are more robust due to the availability of the multiple paths between the source and the destination.

Tree based multicast protocols can be further divided into two types: **source-tree-based** and **shared-tree-based**. In source-tree-based multicast protocols, the tree is rooted at the source, whereas in shared-tree-based multicast, a single tree is shared by all the sources within the multicast group and is rooted at a node referred to as the core bode. The source-tree-based multicast protocols perform better than the shared-tree based protocols at heavy loads because of efficient traffic distribution. But the latter types of protocols are more scalable. The main problem in shared-tree-based multicast protocols is that it heavily depends on the core node and hence a single point of failure at the core node affects the performance of the multicast protocol.

2.4.2 Based on initialization of multicast session

The multicast group formation can be initiated by the source as well as the receivers. In a multicast protocol, if the group formation is initiated only be the source node, then it is called a **source-initiated** multicast routing protocol, and if it is initiated by the receivers of the multicast group then it is called a receiver-initiated multicast routing protocol. Some multicast protocols do not distinguish

between source and receiver for initialization of the multicast group. We call these protocols as source or **receiver-initiated** multicast routing protocols.

2.4.3 Based on topology maintenance mechanism

Maintenance of the multicast topology can be done either by the **soft state** approach or by the **hard state** approach. In the soft state approach, control packets are flooded periodically to refresh the route, which leads to a high packet delivery ratio at the cost of more control packets. Where in case of hard state approach packets are transmitted only when a link breaks, resulting in lower control overhead, but at the cost of a low packet delivery ratio.

3 MULTICAST ROUTING OVER TREES

With few exceptions, the methods used today for supporting many-to-many communication (multicasting) efficiently in computer networks involve routing trees. The basic approach consists of establishing a routing tree for a group of routing nodes (routers). Once a routing tree is established for a group of routers, a packet or message sent to all the routers in the tree traverses each router and links in the tree only once. Multicast routing trees (multicast trees for short) are being used extensively for multicast routing in computer networks and internets, and have also been proposed for wireless multi-hop networks [5].

Because it provides a single path between any two participating routers, multicast trees generate the minimum number of copies per packet to disseminate packets to all the receivers in a multicast group. For a tree of N routers, only N - 1 links are used to transmit the same information to all the routers in the multicast tree in a network with point-to-point links; in the case of wireless networks with broadcast links using a single channel, each member of a multicast tree needs to transmit a packet only once [4]. Using routing trees is of course far more efficient than the brute-force approach to send the same information from the source individually to each of the other N - 1 node. An additional benefit of using trees for multicast routing is that the routing decisions at each router in the multicast tree become very simple: a router in a multicast tree that receives a multicast packet for the group over an in-tree interface forwards the packet over the rest of its in-tree interfaces.

However, multicast trees achieve the efficiency and simplicity described above by forcing a single path between any pair of routers. Accordingly, if multiple sources must transmit information to the same set of destinations, using routing trees require that either a shared multicast tree be used for all sources, or that a separate multicast tree be established for each source. Using a shared multicast tree has the disadvantage that packets are distributed to the multicast group along paths that can be much longer than the shortest paths from sources to receivers [5]. Using a separate multicast tree for each sources of multicast group force the routers, which participate in multiple multicast groups to maintain an entry for each source in the multicast group, which does not scale as the number of groups

and sources per group increase [6]. In addition, because trees provide minimal connectivity among the member of a multicast group, the failure of any link in the tree partitions the group and requires the routers involved to reconfigure the tree.

An ad-hoc network is a packet-switching network based on wireless links for router interconnection. The topology of an ad-hoc network can be very dynamic due to the mobility of routers and the characteristics of the radio channels. Tree-based multicast routing is very attractive for wired networks and the Internet because of its simplicity, but the performance of tree based protocols starts degrading as the mobility increases [4, 6].

This chapter focuses on multicast communication in ad-hoc networks and presents an improvement in the MAODV protocol under the high mobility scenario, when nodes in the network move frequently. The key contributions of this chapter consist of proving that it is possible to improve the performance of the tree based protocol under high mobility scenario.

3.1 OVERVIEW OF MAODV

The multicast operation of the Ad hoc On-Demand Distance Vector (AODV) routing protocol (MAODV) is intended for use by mobile nodes in an ad hoc network. It offers quick adaptation to dynamic link conditions, low processing and memory overhead, and low network utilization. It creates bi-directional shared multicast trees connecting multicast sources and receivers. These multicast trees are maintained as long as group members exist within the connected portion of the network. Each multicast group has a group leader whose responsibility is maintaining the group sequence number, which is used to ensure freshness of routing information.

The Multicast Ad hoc On-Demand Distance Vector (MAODV) protocol enables dynamic, self-starting, multi-hop routing between participating mobile nodes wishing to join or participate in a multicast group within an ad hoc network [7]. The membership of these multicast groups is free to change during the lifetime of the network. MAODV enables mobile nodes to establish a tree connecting multicast group members. Mobile nodes are able to respond quickly to link breaks in

multicast trees by repairing these breaks in a timely manner. In the event of a network partition, multicast trees are established independently in each partition, and trees for the same multicast group are quickly connected if the network components merge.

One distinguishing feature of MAODV is its use of sequence numbers for multicast groups. Each multicast group has its own sequence number, which is initialized by the multicast group leader and incremented, periodically [7]. Using these sequence numbers ensures that routes found to multicast groups are always the most current ones available. Given the choice between two routes to a multicast tree, a requesting node always selects the one with the greatest sequence number. MAODV is the multicast protocol associated with the Ad hoc On-Demand Distance Vector (AODV) routing protocol, and as such it shares many similarities and packet formats with AODV. The Route Request and Route Reply packet types are based on those used by AODV, as is the unicast Route Table. Similarly, many of the configuration parameters used by MAODV are defined by AODV.

3.3.1 MAODV Terminology

Active route

Active route is a route towards a destination that has a routing table entry that is marked as valid. Only active routes can be used to forward data packets.

Broadcast

Broadcasting means transmitting to the IP Limited Broadcast address, 255.255.255.255. A broadcast packet may not be blindly forwarded, but broadcasting is useful to enable dissemination of MAODV messages throughout the ad hoc network.

Destination

Destination is an IP address to which data packets are to be transmitted. Same as "destination node". A node knows it is the destination node for a typical data packet when its address appears in the appropriate field of the IP header. Routes for destination nodes are supplied by action of the AODV protocol, which carries the IP address of the desired destination node in route discovery messages.

Forwarding node

A node that agrees to forward packets destined for another node, by retransmitting them to a next hop that is closer to the unicast destination along a path that has been set up using routing control messages.

Forward route

A route set up to send data packets from a node originating a Route Discovery operation towards its desired destination.

Invalid route

Invalid route is a route that has expired, denoted by a state of invalid in the routing table entry. An invalid route is used to store previously valid route information for an extended period of time. An invalid route cannot be used to forward data packets, but it can provide information useful for route repairs, and also for future RREQ messages.

Originating node

Originating node is a node that initiates an AODV route discovery message to be processed and possibly retransmitted by other nodes in the ad hoc network. For instance, the node initiating a Route Discovery process and broadcasting the RREQ message is called the originating node of the RREQ message.

Reverse route

A route set up to forward a reply (RREP) packet back to the originator from the destination or from an intermediate node having a route to the destination.

Sequence number

A monotonically increasing number maintained by each originating node. In AODV routing protocol messages, it is used by other nodes to determine the freshness of the information contained from the originating node.

Valid route

It is same as active route.

Group leader

Group leader is a node which is a member of the given multicast group and which is typically the first such group member in the connected portion of the network. This node is responsible for initializing and maintaining the multicast group destination sequence number.

Group leader table

It is a table where ad hoc nodes keep information concerning each multicast group and its corresponding group leader. There is one entry in the table for each multicast group for which the node has received a Group Hello message.

Multicast tree

It is a tree containing all nodes which are members of the multicast group and all nodes which are needed to connect the multicast group members.

Multicast route table

It is a table where ad hoc nodes keep routing (including next hops) information for various multicast groups.

Reverse route

A route set up to forward a reply (RREP) packet back to the source from the destination or from an intermediate node having a route to the destination.

3.2 MAODV OPERATIONS

This section describes the scenarios under which nodes generate control messages for multicast communication, and how the fields in the messages are handled [7].

3.2.1 Generating Route Requests

A node sends a RREQ either when it determines that it should be a part of a multicast group, and it is not already a member of that group, or when it has a message to send to the multicast group but does not have a route to that group. If the node wishes to join the multicast group, it sets the 'J' flag in the RREQ; otherwise, it leaves the flag unset. The destination address of the RREQ is always

set to the multicast group address. If the node does not have a route to the group leader, or if it does not know who the multicast group leader is, it broadcasts the RREQ and does not include the extension field. After transmitting the RREQ, the node waits for the reception of a RREP. The node may resend the RREQ up to RREQ_RETRIES additional times if a RREP is not received. If a RREQ was unicast to a group leader and a RREP is not received within RREP_WAIT_TIME milliseconds, the node broadcasts subsequent RREQs for that multicast group across the network. If a RREP is not received after RREQ_RETRIES additional requests, the node may assume that there are no other members of that particular group within the connected portion of the network. If it wanted to join the multicast group, it then becomes the multicast group leader for that multicast group and initializes the sequence number of the multicast group. Otherwise, if it only wanted to send packets to that group without actually joining the group, it drops the packets it had for that group and aborts the session.

Figure 3.1 summarized the generating RREQ operation using flow charts. When the node wishes to join or send a message to a multicast group, it first consults its Group Leader Table. Based on the existence of an entry for the multicast group in this table, the node then formulates and sends the RREQ as described at the beginning of this section.

3.2.2 Controlling Route Request Broadcasts

To prevent unnecessary network-wide broadcasts of RREQs, the source node should use an expanding ring search technique. When a RREP is received, the Hop Count to the group leader used in the RREP packet is remembered as Last Hop Count for that node in the routing table. When a new route to the same multicast group is required at a later time (e.g., upon a link break in the multicast tree), the TTL in the RREQ IP header is initially set to this Last Hop Count plus TTL_INCREMENT. Thereafter, following each timeout the TTL is incremented by TTL_INCREMENT until TTL = TTL_THRESHOLD is reached.

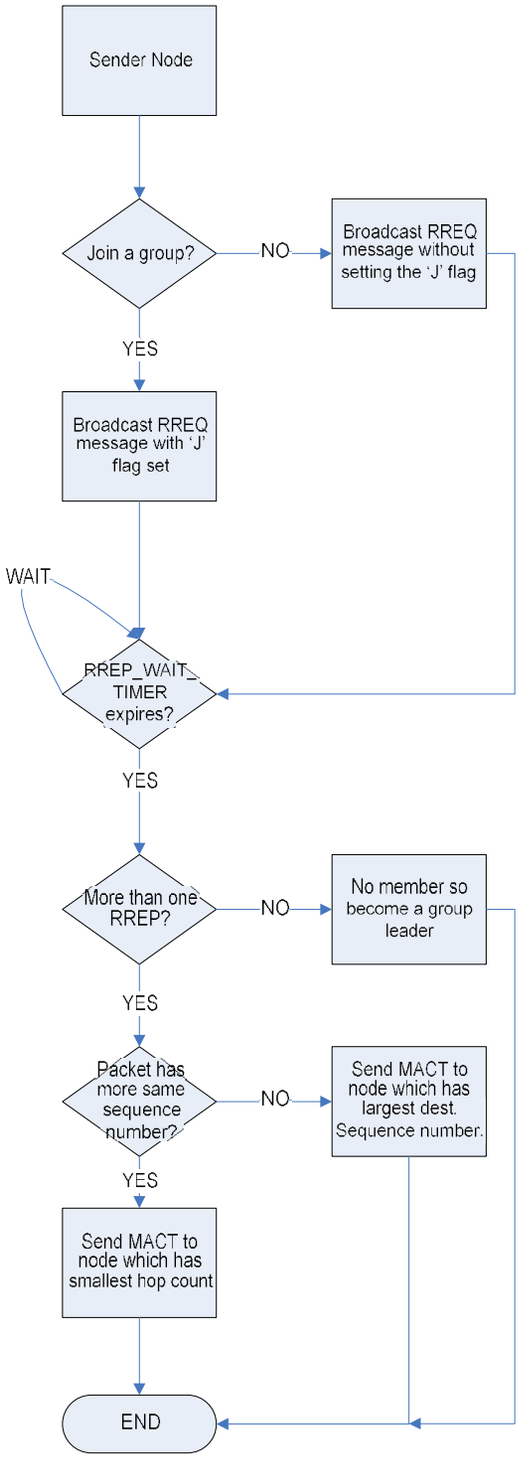


Figure 3.1 Generating RREQ

3.2.3 Receiving Route Requests

When a node receives a RREQ, the node checks whether the 'J' flag of the RREQ is set. If the 'J' flag is set, the node can only respond if it is a member of the multicast tree for the indicated multicast group, and if its record of the multicast

group sequence number is at least as great as that contained in the RREQ. If the 'J' flag is not set, then the node can respond if it has an unexpired route to the multicast group and the above multicast group sequence number criteria is met.

If the node does not meet either of these conditions, it rebroadcasts the RREQ from its interface(s) but using its own IP address in the IP header of the outgoing RREQ. The Destination Sequence Number in the RREQ is updated to the maximum of the existing Destination Sequence Number in the RREQ and the multicast group sequence number in the multicast route table (if an entry exists) of the current node. The TTL or hop limit field in the outgoing IP header is decreased by one. The Hop Count field in the broadcast RREQ message is incremented by one, to account for the new hop through the intermediate node. The node always creates or updates a reverse route. This reverse route would be needed in case the node receives an eventual RREP back to the node which originated the RREQ (identified by the Source IP Address).

In addition to creating or updating the route table entry for the source node, the node receiving the RREQ also creates a next hop entry for the multicast group in its multicast route table. If no entry exists for the multicast group, it creates one, and then places the node from which it received the RREQ as next hop for that group. It leaves the Activated flag associated with this next hop unset. The direction for this next hop entry is downstream.

3.2.4 Generating Route Reply

If a node receives a join RREQ for a multicast group, and it is already a member of the multicast tree for that group, the node updates its multicast route table and then generates a RREP message. The Source and Destination IP Addresses in RREQ message are copied to corresponding fields in the RREP message. The RREP contains the current sequence number for the multicast group and the IP address of the group leader. Furthermore, the node initializes the Hop Count field of the RREP to zero. It unicasts the RREP back to the node indicated by the Source IP Address field of the received RREQ.

A node can respond to a join RREQ only if it is a member of the multicast tree. If a node receives a multicast route request that is not a join message, it can reply if it has a current route to the multicast tree. Otherwise it continues forwarding the request. If a node receives a join RREQ for a multicast group and it is not already a member of the multicast tree for that group, it rebroadcasts the RREQ to its neighbors.

When a node receives a RREQ for a multicast group which has its own IP address in the Destination IP address of the IP header, that means that the source node expects this destination node to be the multicast group leader. In this case, if the node is in fact not the group leader, it can simply ignore the RREQ. The source node will time out after RREP_WAIT_TIME milliseconds and will broadcast a new RREQ without the group leader address specified.

3.2.5 Forwarding Route Replies

If an intermediate node receives a RREP in response to a RREQ that it has transmitted (or retransmitted on behalf of some other node), it creates a multicast group next hop entry for the node from which it received the RREP. The direction of this next hop is upstream, and the Activated flag is left unset. Additionally, the node updates the Lifetime field in the route table entry associated with the node from which it received the RREP. It then increments the Hop Count and Multicast Group Hop Count fields or the RREP and forwards this packet along the path to the source of the RREQ.

When the node receives more than one RREP for the same RREQ, it saves the route information with the greatest sequence number, and beyond that the lowest hop count; it discards all other RREPs. This node forwards the first RREP towards the source of the RREQ, and then forwards later RREPs only if they have a greater sequence number or smaller metric.

3.2.6 Route Activation

When a node broadcasts a RREQ message, it is likely to receive more than one reply since any node in the multicast tree can respond. The RREP message sets up

route pointers as it travels back to the source node. If the request is a join request, these route pointers may eventually graft a branch onto the multicast tree. Also, because multicast data packets may be transmitted as broadcast traffic, the route to the multicast tree must be explicitly selected. Otherwise, each node with a route to the tree which receives a multicast data packet will rebroadcast the packet, resulting in an inefficient use of network bandwidth. Hence, it is necessary to activate only one of the routes created by the RREP messages. The RREP containing the largest destination sequence number is chosen to be the branch added to the multicast tree (or the path to the multicast tree, if the request was a non-join). In the event that a node receives more than one RREP with the same (largest) sequence number, it selects the first one with the smallest hop count, i.e., the shortest distance to a member of the multicast tree.

After waiting RREP_WAIT_TIME milliseconds, the node must select the route it wishes to use as its link to the multicast tree. This is accomplished by sending a Multicast Activation (MACT) message. The Destination IP Address field of the MACT packet is set to the IP address of the multicast group. If the node is joining the multicast group, it sets the join flag of this message. The node unicasts this message to the selected next hop, effectively activating the route. It then sets the Activated flag in the next hop Multicast Route Table entry associated with that node. After receiving this message, the node to which the MACT was sent activates the route entry for the link in its multicast route table, thereby finalizing the creation of the tree branch. All neighbors not receiving this message time out and delete that node as a next hop for the multicast group in their route tables, having never activated the route entry for that next hop.

Two scenarios exist for a neighboring node receiving the MACT message. If this node was previously a member of the multicast tree, it does not propagate the MACT message any further. However, if the next hop selected by the source node's MACT message was not previously a multicast tree member, it will have propagated the original RREQ further up the network in search of nodes which are tree members. Thus it is possible that this node also received more than one RREP.

Figure 3.2 shows summarized the route activation operation. When the node receives a MACT selecting it as the next hop, it unicasts its own MACT to the node it has chosen as its next hop, and so on up the tree, until a node which was already a part of the multicast tree is reached.

3.2.7 Multicast Tee Pruning

A multicast group member can revoke its member status at any time. However, it can only actually leave the multicast tree if it is not a tree router for any other nodes in the multicast group (i.e., if it is a leaf node). If a node wishing to leave the multicast group is a leaf node, it unicasts to its next hop on the tree a MACT message with the 'P' flag set and with the Destination IP Address set to the IP address of the multicast group. It then deletes the multicast group information for that group from its multicast route table.

When its next hop receives this message, it deletes the sending node's information from its list of next hops for the multicast tree. If the removal of the sending node causes this node to become a leaf node, and if this node is also not a member of the multicast group, it may in turn prune itself by sending its own MACT message up the tree.

When the multicast group leader wishes to leave the multicast group, it proceeds in a manner similar to the one just described. If it is a leaf node, it may leave the group and unicast a prune message to its next hop. Otherwise, if the group leader is not a leaf node, it may not prune itself from the tree. where it selects one of its next hops and unicasts to it the MACT with set 'G' flag.

3.2.8 Repairing A Broken Link

Branches of the multicast tree become invalid if a broken link results in an infinite metric being associated with the next hop route table entry. When a broken link is detected between two nodes on the multicast tree, the two nodes should delete the link from their list of next hops for the multicast group. The node downstream of the break (i.e., the node which is further from the multicast group leader) is responsible for initiating the repair of the broken link.

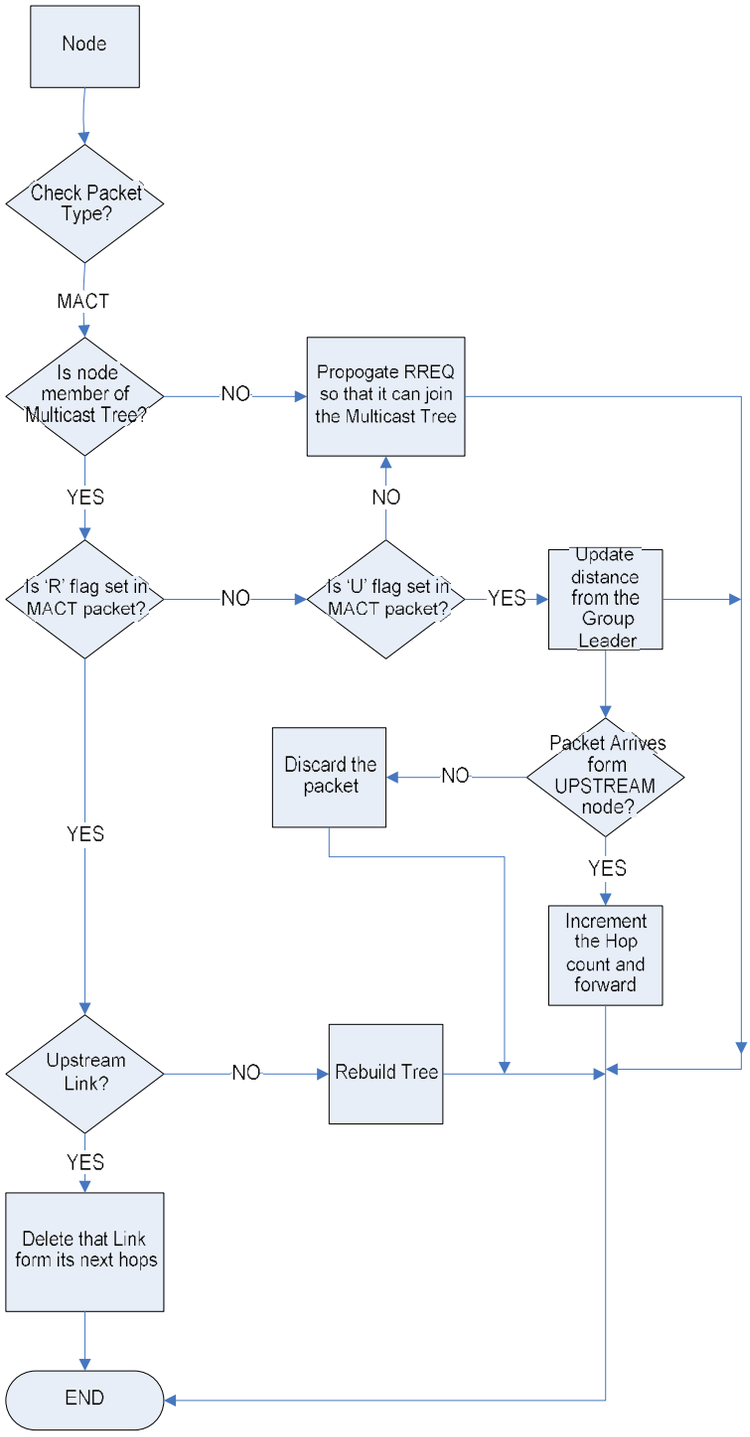


Figure 3.2 Receiving MACT

In order to repair the tree, the downstream node broadcasts a RREQ with destination IP address set to the IP address of the multicast group and with the 'J' flag set. The destination sequence number of the RREQ is the last known sequence number of the multicast group. The node also includes the Multicast Group Leader

Extension. The Multicast Group Hop Count field of this extension is set to the distance of the source node from the multicast group leader. A node must have a hop count to the multicast group leader less than or equal to the indicated value in order to respond. This hop count requirement prevents nodes on the same side of the break as the node initiating the repair from replying to the RREQ.

The RREQ is broadcast using an expanding rings search. Because of the high probability that other nearby nodes can be used to rebuild the route, the original RREQ is broadcast with a TTL (time to live) field value equal to TTL_INCREMENT plus the Multicast Group Hop Count. In this way, the effects of the link breakage may be localized. If no reply is received within RREP_WAIT_TIME milliseconds, the node should increment the TTL of each successive RREQ by TTL_INCREMENT, until either a route is determined, or TTL_THRESHOLD is reached. After this point, if a route is still not discovered, each additional RREQ is broadcast with TTL = NET_DIAMETER. Up to RREQ_RETRIES additional broadcasts may be attempted after TTL = NET_DIAMETER is reached.

A node receiving this RREQ can respond if it meets the following conditions:

- It is a member of the multicast tree.
- Its record of the multicast group sequence number is at least as great as that contained in the RREQ.
- Its hop count to the multicast group leader is less than or equal to the contained in the Multicast Group Hop Count extension field.

At the end of the discovery period, the node selects its next hop and unicasts a MACT message to that node to activate the link. Since the node was repairing a tree break, it is likely that it is now a different distance from the group leader than it was before the break. If this is the case, it must inform its DOWNSTREAM next hops of their new distance from the group leader. It does this by broadcasting a MACT message with the 'U' flag set, and the Hop Count field set to the node's new distance from the group leader. This 'U' flag indicates that multicast tree nodes should update their distance from the group leader. When a node on the multicast

tree receives the MACT message with the 'U' flag set, it determines whether this packet arrived from its upstream neighbor. If it did not, the node discards the packet.

Otherwise, it increments the Hop Count value contained in the MACT packet and updates it distance to the group leader to be this node value. If this node has one or more downstream next hops, it in turn must send a MACT message with a set 'U' flag to its next hops, and so on. Figure 3.2 summarized the operation using flow chart.

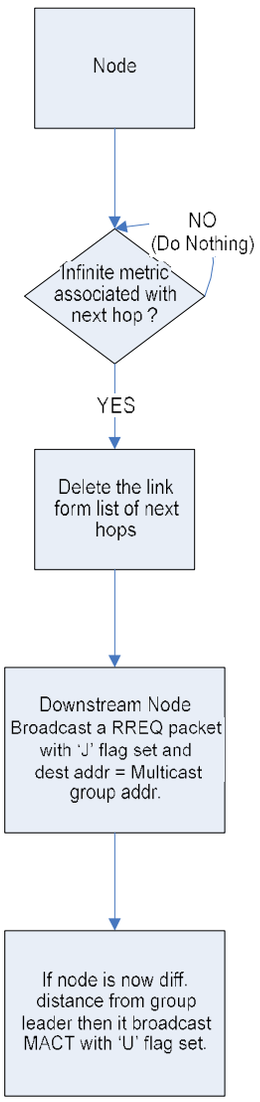


Figure 3.3 repairing a broken link

When a link break occurs, it is possible that the tree will be repaired through different intermediate nodes. If the node upstream of the break is not a group member, and if the loss of that link causes it to become a leaf node, it sets a prune timer to wait for the link to be repaired. This PRUNE_TIMEOUT should be larger than RREP_WAIT_TIMEOUT to give the link time to be repaired. If, when this timer expires, the node has not received a MACT message selecting it to be a part of the repaired tree branch; it prunes itself from the tree by sending a MACT with set 'P' flag to its next hop.

3.2.9 Group Hello Messages

If a node sends a RREQ to join a multicast group (`J' flag set) and after RREQ_RETRIES attempts do not receives a response, it then becomes the multicast group leader. The node initializes the multicast group sequence number and then broadcasts a GRPH message to inform network nodes that it is now the group leader for the multicast group. To ensure nodes maintain consistent and up-to-date information about who the multicast group leaders are, any node which is a group leader for a multicast group broadcasts such a GRPH across the network every GROUP_HELLO_INTERVAL milliseconds.

Nodes receiving the GRPH increment the Hop Count field by one before forwarding the message. When a node not on the multicast tree receives the GRPH message, it sets the 'M' flag. This indicates that this incarnation of the message has traveled off the multicast tree, and hence cannot be used by group members to verify their distance from the group leader. The 'U' flag is set by the group leader whenever there has been a change in group leader information. It informs nodes that they should update the group leader information associated with the indicated multicast group.

A node receiving the GRPH message first verifies that is has not already received the GRPH message with the same group IP address/group sequence number pair in the last BCAST_ID_SAVE milliseconds. If it has, it silently discards the message. Otherwise, the node updates its group leader table to reflect the current multicast group IP address/group leader IP address combination, and increments the Hop Count value in the message. If the node receiving the GRPH message is a member

of the multicast tree, it also updates this information in its multicast route table. If the 'M' flag is unset, then this node can also use the Hop Count value to verify its current distance from the group leader. After processing the GRPH message, the node buffers the group IP address/ group sequence number combination to avoid multiple processing of the same GRPH message. It then rebroadcasts the message to its neighbors.

3.2.10 Tree Partition

It is possible that after a link breaks, the tree cannot be repaired due to a network partition. If the node attempting to repair a tree link break does not receive a response after RREQ_RETRIES attempts, it can be assumed that the network has become partitioned and the multicast tree cannot be repaired at this time. In this situation, if the node which initiated the route rebuilding is a multicast group member, it becomes the new multicast group leader for its part of the multicast tree partition. It increments the group sequence number and then broadcasts a Group Hello (GRPH) for this multicast group. Te 'U' flag in the GRPH is set, indicating that there has been a change in the group leader information. All nodes receiving this message update their group leader table to indicate the new group leader information. Nodes which are a part of the multicast tree also update the group leader and sequence number information for that group in their multicast route table. On the other hand, if the node which had initiated the repair is not a multicast group member, there are two possibilities. If it only has one next hop for the multicast tree, it prunes itself from the tree by unicasting a MACT message, with the 'P' flag set, to its next hop. The node receiving this message notes that the message came from its upstream link, i.e., from a node that is closer to the group leader than it is. If the node receiving this message is a multicast group member, it becomes the new group leader and broadcasts a GRPH message as indicated above. Otherwise, if it is not a multicast group member and it only has one other next hop link, it similarly prunes itself from the tree. This process continues until a multicast group member is reached.

The other possibility is that the node which initiated the rebuilding is not a group member and has more than one next hop for the tree. In this case, it cannot prune itself, since doing so would partition the tree. It instead selects one of its next

hops and unicasts a MACT with the 'G' flag set to that node. This flag indicates that the next group member to receive this message should become the new group leader. It then changes the direction of that link to be UPSTREAM. If the node receiving the MACT is a group member, then this node becomes the new group leader. Otherwise, the node unicasts its own MACT message with the 'G' flag set to one of its next hops, and changes the direction of that link. Once a group member is reached, the new group leader is determined.

3.2.11 Tree Merge

In the event that a link break cannot be repaired, the multicast tree remains partitioned until the two parts of the network become connected once again. A node from one partition of the network knows that it has come into contact with a node from the other partition of the network by noting the difference in the GRPH message multicast group leader information. The multicast group leader with the lower IP address initiates the tree repair. For the purposes of this explanation, call this node GL1. GL1 unicasts a RREQ with both the 'J' and 'R' flags set to the group leader of the other network partition (GL2), using the node from which it received the GRPH as the next hop. This RREQ contains the current value of GL1's multicast group sequence number. If any node that receives the RREQ is a member of GL2's multicast tree, it MUST forward the RREQ along its upstream link, i.e. towards GL2. This prevents any loops from being formed after the repair. Upon receiving the RREQ, GL2 takes the larger of its and the received multicast group sequence number, increments this value by one, and responds with a RREP. This is the group leader which becomes the leader of the reconnected multicast tree. The 'R' flag of the RREP is set, indicating that this RREP is in response to a repair request.

As the RREP is propagated back to GL1, nodes add the incoming and outgoing links to the multicast route table next hop entries if these entries do not already exist. The nodes also activate these entries, thereby adding the branch on to the multicast tree. If a node that was previously a member of GL1's tree receives the RREP, it must forward the packet along its link to its previous group leader (GL1). It then updates its group leader information to reflect GL2 as the new group leader, changes the direction of the next hop link associated with GL1 to

DOWNSTREAM, and sets the direction of the link on which it received the RREP to UPSTREAM. When GL1 receives the RREP, it updates its group leader information and sets the link from which it received the RREP as its upstream link. The tree is now reconnected. The next time GL2 broadcasts a GRPH, it sets the `U' flag to indicate that there is a change in the group leader information and group members should update the corresponding information. All network nodes update their group leader table to reflect the new group leader information

3.2.12 Actions After Reboot

A node participating in the multicast tree that reboots (or restarts the routing daemon) loses all of its multicast tree information. Upon reboot, the rebooted node does not know whether it was previously a member of the multicast tree. Hence, it should unconditionally broadcast a MACT message with set Reboot ('R') flag to inform neighboring nodes that it has lost its multicast group information. When a node on the multicast tree receives the reboot MACT message, it checks whether this message came from one of its next hops on the multicast tree. If so, one of two situations exists.

If the reboot MACT came from a downstream link, the node deletes that link from its list of next hops and sets a prune timer. Otherwise, if the reboot MACT came from a node's upstream link, it must rebuild the tree branch. Figure 3.3 shows action of a node after it reboot.

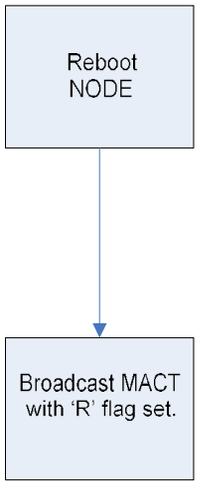


Figure 3.3 Actions after Reboot

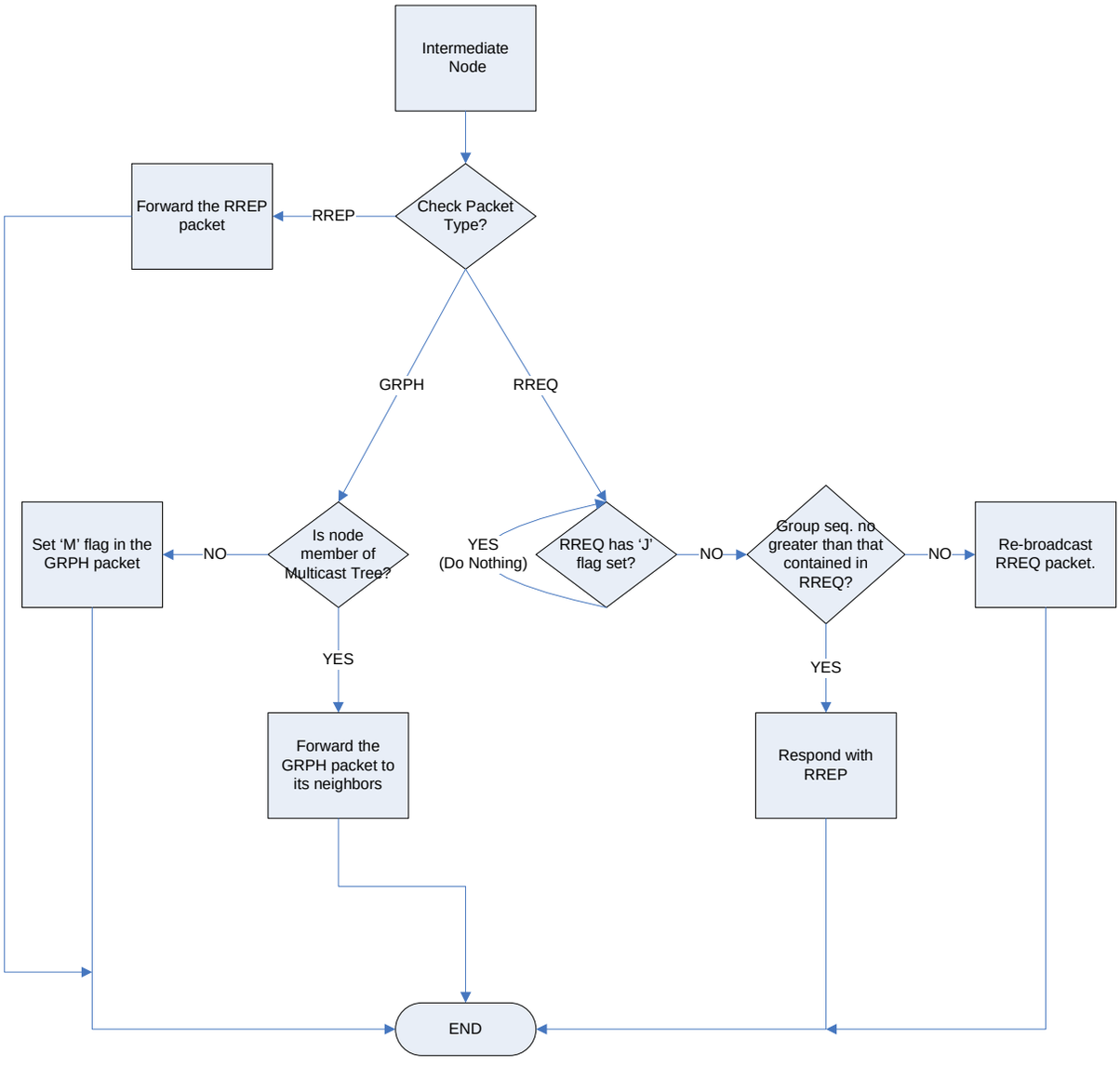


Figure 3.5 Receiving packets (Intermediate node)

3.2.13 Maintaining Local Connectivity

Each node on the multicast tree must keep track of its neighbors which are next hops on the multicast tree. If a link to the next hop cannot be detected by any of the methods specified therein, the forwarding node must assume that the link is broken and try to repair the link.

Whenever the multicast tree is used for the transmission of data packets, a node on the tree must hear from each of its next hop neighbors every RETRANSMIT_TIME. The node can hear from its neighbors through one of the methods previously described, or through the reception of a rebroadcast of a data

packet from these next hops, if the data packets are being transmitted by broadcast. If a node does not hear from one of its multicast tree next hops within RETRANSMIT_TIME, and if the multicast tree is actively being used to transmit data packets, then the node must assume the link to its next hop is broken.

Figure 3.5 and 3.6 shows the operation of the intermediate and receiver node respectively when they receive different packets

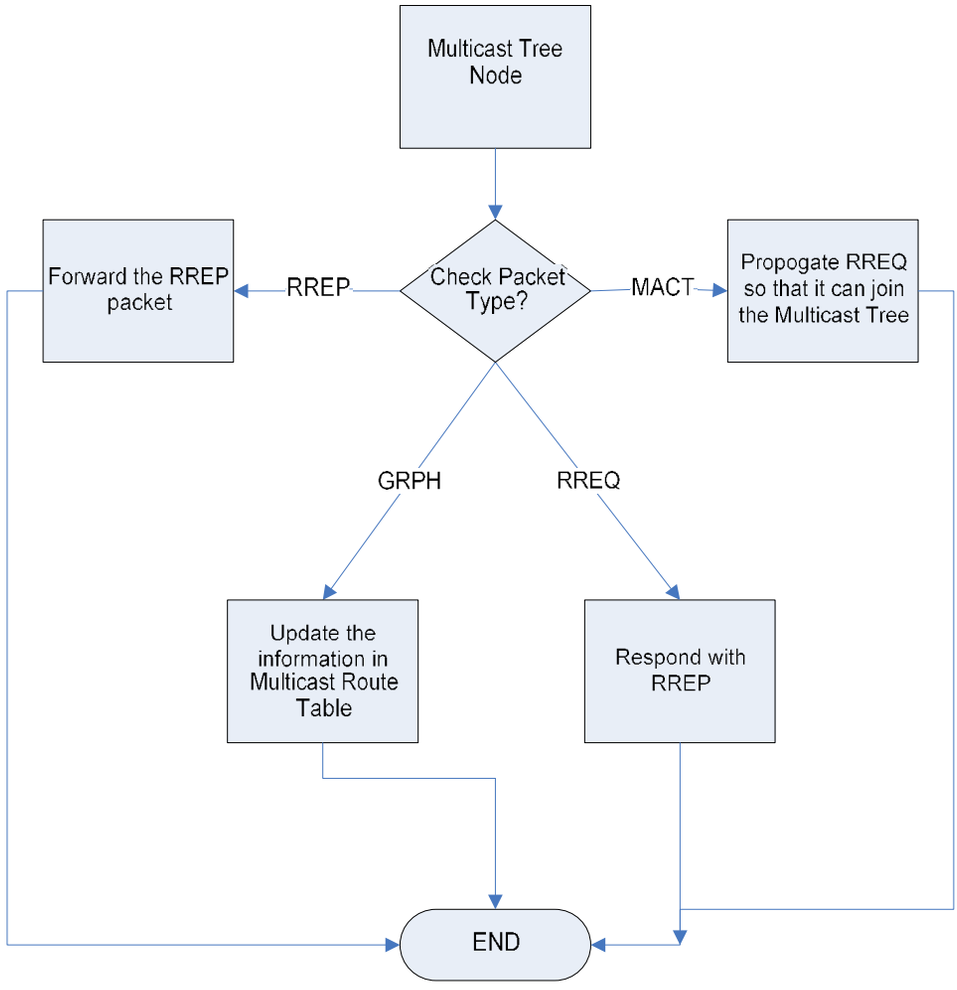


Figure 3.6 receiving Packets (Receiving nodes)

The above operations are the basic MAODV operations. To improve the performance of the MAODV protocol under high mobility scenario, where node are moving frequently, I have included new packet called RI (Receiver Identification) packet. The following operation describes when to send this packet and what a node will do when they receive RI packets.

3.2.14 Generating Receiver Identification

Every multicast tree has a group member. Only sender can generate RI (Receiver Identification) packets every few seconds. Sender broadcast this packet with 'J' flag set to all its neighbors and waits for the RI_TIMEOUT timer for the reply of RI packet. Sender generates this packet to detect for the receiver, if receiver is within its range of the sender then sender directly unicast data packets to receiver rather than go through intermediate nodes.

3.2.15 Generating Receiver Identification Reply

If a node receives a join RI packet and it is already a member of the multicast tree for that group, then node checks the destination address of the packet, if it is same as its address (which indicates that now it is within the range of the sender), so it replies with RI packet without setting 'J' flag.

If a node receives a join RI packet, and it is not a group member or the destination address is not its address then it just discard the packet. Now when a sender receives a RI packet with 'J' flag not set, it indicates sender that now receiver is within its range, so now rather than forwarding packets through intermediate nodes its directly unicast it to receiver. Sender also send RI packet with 'P' flag set to that node through which it was sending the packets. Now the node receiving that packet from sender node deletes the routing table entry for the receiver node and forwards this packet along the path so that all intermediate nodes delete the entry for the receiver node.

4 STUDY OF PETRI NETS

4.1 INTRODUCTION

A Petri Net is a particular kind of directed graph, together with an initial state called the initial marking, M_0. The underlying graph N of a Petri net is a directed, weighted, bipartite graph consisting of two kinds of nodes, called places (resources) and transitions (node which consumes resources), where arcs are either from a place to a transition or from a transition to a place [8].

In graphical representation, places are drawn as circles, transitions as bars or boxes. Arcs are labeled with their weights (positive integers), where a k-weighted arc can be interpreted as the set of k parallel arcs. Labels for unity weight are usually omitted. A marking (state) assigns to each place a nonnegative integer. If a marking assigns to place p a nonnegative integer k, we say that p is marked with k tokens. Pictorially, we place k black dots (tokens) in place p. A marking is denoted by M, an m-vector, where m is the total number of places. The p^{th} component of M, denoted by M(p), is the number of tokens in place p [8].

In modeling, using the concept of conditions and events, places represent conditions, and transitions represent events. A transition (an event) has a certain number of input and output, places representing the pre-conditions and post-conditions of the event, respectively. The presence of a token in a place is interpreted as holding the truth of the condition associated with the place. In another interpretation, k tokens are put in a place to indicate that k data items or resources are available.

A Petri net is a 5-tuple PN = (P, T, Pre, Post; M) where P is a set of places (represented by circles), T is a set of transitions (represented by bars), Pre: P X T -> (0, l} is the forward incidence function (represented by an arc from place p to transition t if Pre (p, t) = l), Post: P X T ->{O, l} is the backward incidence function (represented by an arc from transition t to place p if Post (p, t) = l), Mo is the initial marking (markings, denoted by M, are mappings of P on the set of natural numbers N, and M(p) = n is represented by n tokens inside the place p).

The behavior of many systems can be described in terms of system states and their changes. In order to simulate the dynamic behavior of a system, a state or marking in a Petri nets is changed according to the following transition (firing) rule [8]:

1. A transition t is said to be enabled if each input place p of t is marked with at least w(p, t) tokens, where w(p,t) is the weight of the arc from p to t.

2. An enabled transition may or may not fire (depending on whether or not the event actually takes place).

3. A firing of an enabled transition t removes w(p, t) tokens from each input place p of t, and adds w(t, p) tokens to each output place p of t, where w(t, p) is the weight of the arc from t to p.

A transition without any input place is called a source transition, and one without any output place is called a sink transition. Note that a source transition is unconditionally enabled, and that the firing of a sink transition consumes tokens, but does not produce any.

4.2 PETRI NET MODEL FOR MAODV

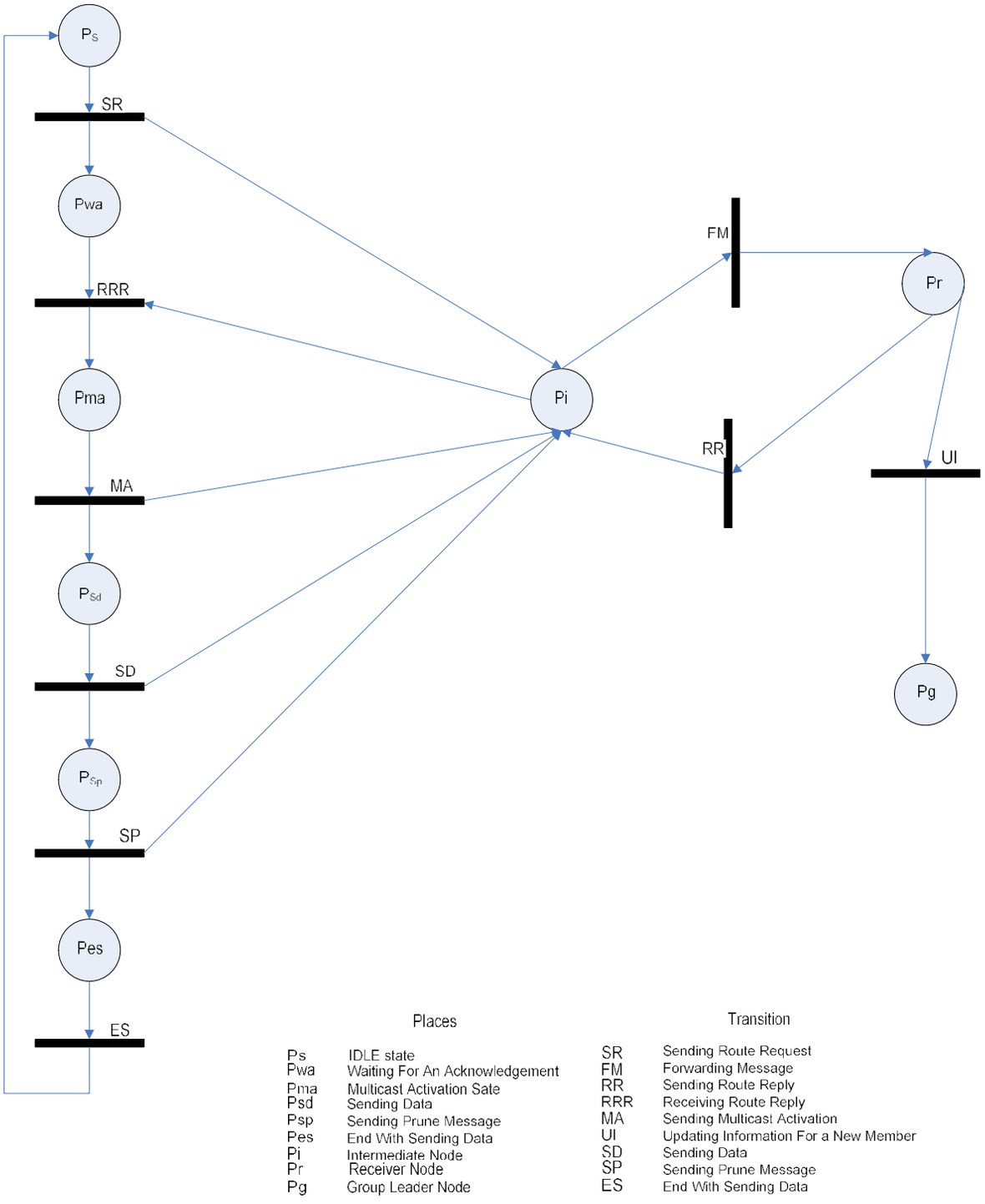


Figure 4.1 Petri Net model for MAODV

4.3 PETRI NET MODEL PROPERTIES

4.3.1 Reachability

Reachability is a fundamental basis for studying the dynamic properties of any system. The firing of an enabled transition will change the token distribution (marking) in a net according to the transition rule [9]. A sequence of firings will result in a sequence of markings. A marking M, is said to be reachable from a marking M_0 if there exists a sequence of firings that transforms M_0 to M,

Here in the Figure 4.1 marking Mn (Last Marking of a model) is reachable form Marking Mo (Initial marking). So we can say that the MAODV protocol is reachable.

4.3.2 Boundedness

A Petri net (N, M_0**)** is said to be k-bounded or simply bounded if the number of tokens in each place does not exceed a finite number k for any marking reachable from M_0**,** i.e., M(p) ≤ k for every place p and every marking M ε R(Mo). A Petri net (N, M_0) is said to be safe if it is 1-bounded [9].

Here (Figure 4.1) maximum number of tokens that a place can have, is depend upon the capacity of the queue but it is finite so we can say that the given model is k-bounded (k = maximum limit of the queue), if we take an idle state where k is equal to 1 then the given model will be 1-bounded and so it is safe.

4.3.3 Liveness

Liveness is an ideal property for many systems. However, it is impractical and too costly to verify this strong property for some systems such as the operating system of a large computer. Thus, we relax the liveness condition and define different levels of liveness as follows [9].

0) dead (L0-live) if t can never be fired in any firing sequence in L (Mo).

1) L1-live (potentially firable) if t can be fired at least once in some firing

sequence in L (M_0).

2) L2-live if, given any positive integer k, t can be fired at least k times in some firing sequence in L(Mo).

3) L3-live if t appears infinitely, often in some firing sequence in L (Mo).

4) L4-liveorliveif t is L1-live for every marking M in R (Mo).

Figure 4.1 shows that every transition is firable in some state of a model so it is not L0-live. We can fire all the transition in some state of the model so the given model is L1-live. Here transition SD (Sending Data) can be fired finite k times in the model so the given model is l2-live. Here there is no transition that can be fired infinite time so the given model is not l3-live and also l4-live.

4.3.4 Reversibility and Home State

A Petri net (N, M_0) is said to be reversible if, for each marking M in R(Mo), M_0 is reachable from M. Thus, in a reversible net one can always get back to the initial marking or state. In many applications, it is not necessary to get back to the initial state as long as one can get back to some (home) state. Therefore, we relax the reversibility condition and define a home state. A marking M' is said to be a home state if, for each marking M in R(Mo), M' is reachable from M [9].

Here the given model is reversible since one can reach to initial state (idle state) after node had sent all its data.

4.3.5 Persistence

A Petri net (N, M_0) is said to be persistent if, for any two enabled transitions, the firing of one transition will not disable the other. A transition in a persistent net, once it is enabled, will stay enabled until it fires [9].

Figure 4.1 shows that in the model with two enabled transitions, firings of one will not disabling the other transition, so transition remains enabled until it fires hence, the model is persistence.

5 SIMULATION ENVIORNMENT

The simulator which has been used to simulate the ad-hoc routing protocols is the Network Simulator from Berkely [10]. To simulate the mobile wireless radio environment we have used the mobility extension to NS that is developed by the CMU monarch project at Carnegie Mellon University [10].

5.1 NETWORK SIMULATOR

Network Simulator 2 is the result of an on-going effort of research and development that is administrated by researchers at Berkely [10]. It is a discrete event simulator targeted at networking research. It provides substantial support for simulation of TCP, routing, and multicast protocols.

The simulator is written in C++ and a script language called OTCL. Ns use an Otcl interpreter towards the user. This means that the user writes an Otcl script that defines the network (number of nodes, links), the traffic in the network (sources, destination, type of traffic) and which protocol it will use. This script is then used by ns during the simulations. The results of the simulations is an output trace file that can be used to do data processing (calculate delay, throughput etc.) and to visualize the simulation with a program called network animator (NAM). NAM is very good visualization tool that visualize the packets as they propagate through network. An overview of how the simulation is done in ns is shown in Figure 5.1.

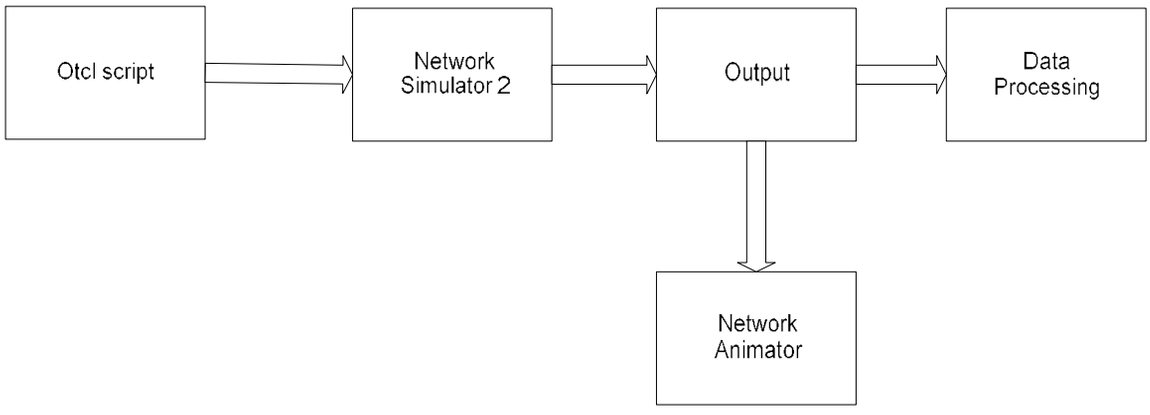


Figure 5.1 Network Simulator 2

The current version of the Network Simulator does not support mobile wireless environments. The Network Simulator alone is only intended for stationary networks with the wired links. This causes some problem problems in the beginning of this master thesis. We needed mobility for our multicasting environment, for that CMU monarch mobility extension to NS is used.

5.2 MOBILITY EXTENSION

This section describes the wireless model that was originally ported as CMU's Monarch group's mobility extension to NS. It covers the original mobility model ported from CMU/Monarch group [10]. The internals of a mobile node, routing mechanisms and network components that are used to construct the network stack for a mobile node are covered. The components that are covered briefly are Channel, Network interface, Radio propagation model, MAC protocols, Interface Queue, Link layer and Address resolution protocol model (ARP).

5.2.1 Network Components Of Mobile Nodes

Link Layer

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.

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. Incase additional packets to the same destination is 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.

Interface Queue

The class PriQueue 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.

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.

Tap Agents

Agents that subclass themselves as class Tap 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 all packets received by the mac layer, before address filtering is done.

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 class Phy/WirelessPhy. 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.

Radio Propagation Model

It uses Friss-space attenuation ($1/r^2$) at near distances and an approximation to two ray Ground ($1/r^4$) at far distances. The approximation assumes specular reflection off a flat ground plane.

Antenna

An omni-directional antenna having unity gain is used by mobile nodes.

Node Mobility

Each mobile node is an independent entity that is responsible for computing its own position and velocity as a function of time. Nodes move around according to a movement pattern specified at the beginning of the simulation.

5.2.2 Mobile Node

Each mobile node (Figure 5.2) makes use of a routing agent for the purpose of calculating routes to other nodes in the ad-hoc network [10]. Packets are sent from the application and are received by the routing agent. The agent decides a path that the packet must travel in order to reach its destination and stamps it with this information. It then sends the packet down to the link layer. The link layer level uses an Address Resolution Protocol (ARP) to decide the hardware addresses of neighboring nodes and map IP addresses to their correct interfaces. When this information is known, the packet is sent down to the interface queue and awaits a signal from the Multiple Access Control (MAC) protocol. When the MAC layer decides it is ok to send it onto the channel, it fetches the packet from the queue and hands it over to the network interface which in turn sends the packet onto the radio channel. This packet is copied and is delivered to all network interfaces at the time at which the first bit of the packet would begin arriving at the interface in a physical system. Each network interface stamps the packet with

the receiving interfaces properties and then invokes the propagation model. The propagation model uses the transmit and receive stamps to determine the power with which the interface will receive the packet. The receiving network interfaces then use their properties to determine if they actually successfully received the packet, and send it to the MAC layer if appropriate. If the MAC layer receives the packet error- and collision- free, it passes the packet to the mobiles entry point. From there it reaches a de-multiplexer, which decides if the packet should be forwarded again, or if it has reached its destination node. If the destination node is reached, the packet is sent to a port de-multiplexer, which decides to what application the packet should be delivered. If the packet should be forwarded again the routing agent will be called and the procedure will be repeated

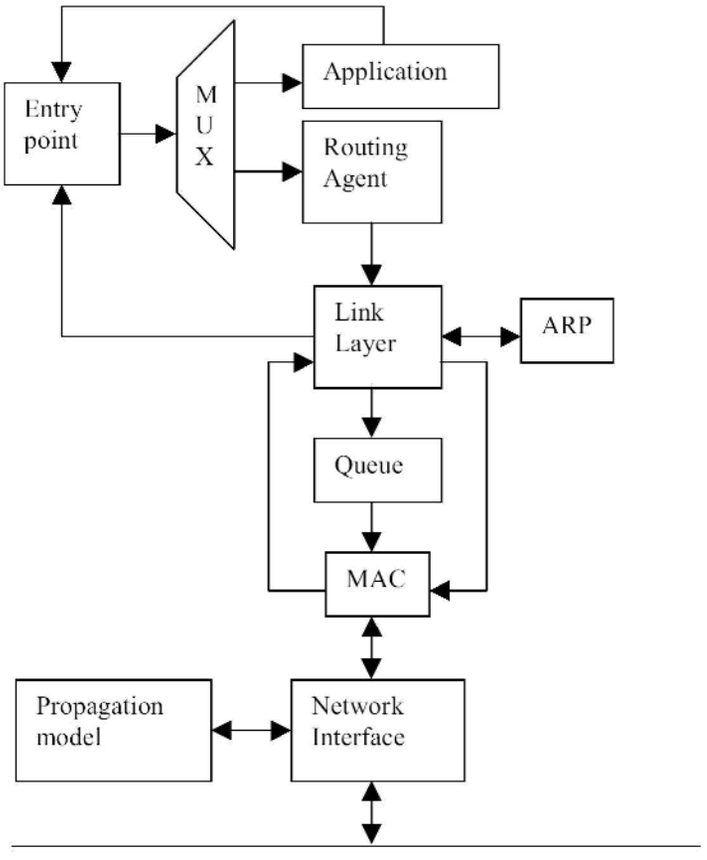


Figure 5.2 A Mobile Node

5.2.3 Simulation Overview

A typical simulation with ns and the mobility extension is shown in Figure 5.3. Basically it consists of generating the following input files to NS: A scenario file that describes the movement pattern of the nodes, communication file that describes the traffic in the network.

These files can be generating completely randomized movement and communication patterns with a script. These files are then used for the simulation and as a result from this, a trace file is generated as output. Prior to the simulation, the parameters that are going to be traced during the simulation must be selected. The trace file can then be scanned and analyzed for the various parameters that we want to measure. This can be used as data for plots with for instance Gnuplot.

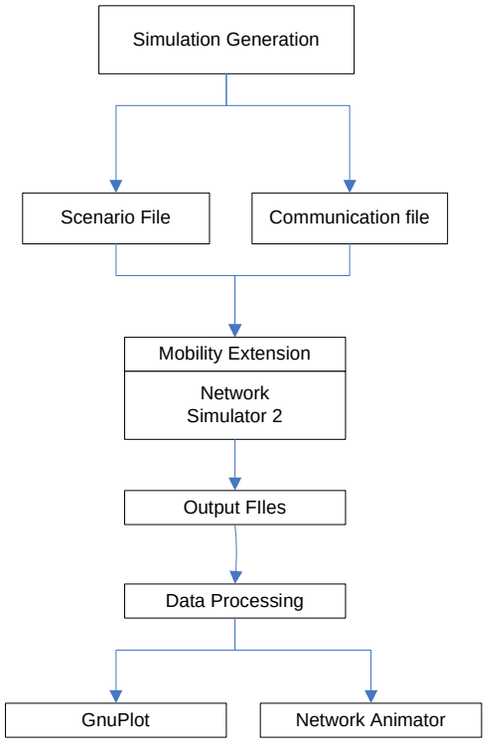


Figure 5.3 Simulation Overview

5.3 TRACE FILES

In an effort to merge wireless trace, using CMU-Trace objects, with ns tracing, a new, improved trace format has been introduced. Currently this new trace support is available for wireless simulations only and shall be extended to rest of ns in the near future. An example of the new trace format is shown below [10].

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

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

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 time

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 energy level
Nl Trace level such as AGT, RTR, MAC

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 Packt 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 towards 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 AODV, MAODV etc being traced.

6.1 INTRODUCTION

Network Simulator 2 tool is used for the simulation of MAODV multicast routing protocol [10]. Now this chapter covers the implementation details of the protocol and how it is implemented in Network Simulator 2. Figure 6.1 shows the basic block diagram of the protocol for simulating it in Network Simulator 2.

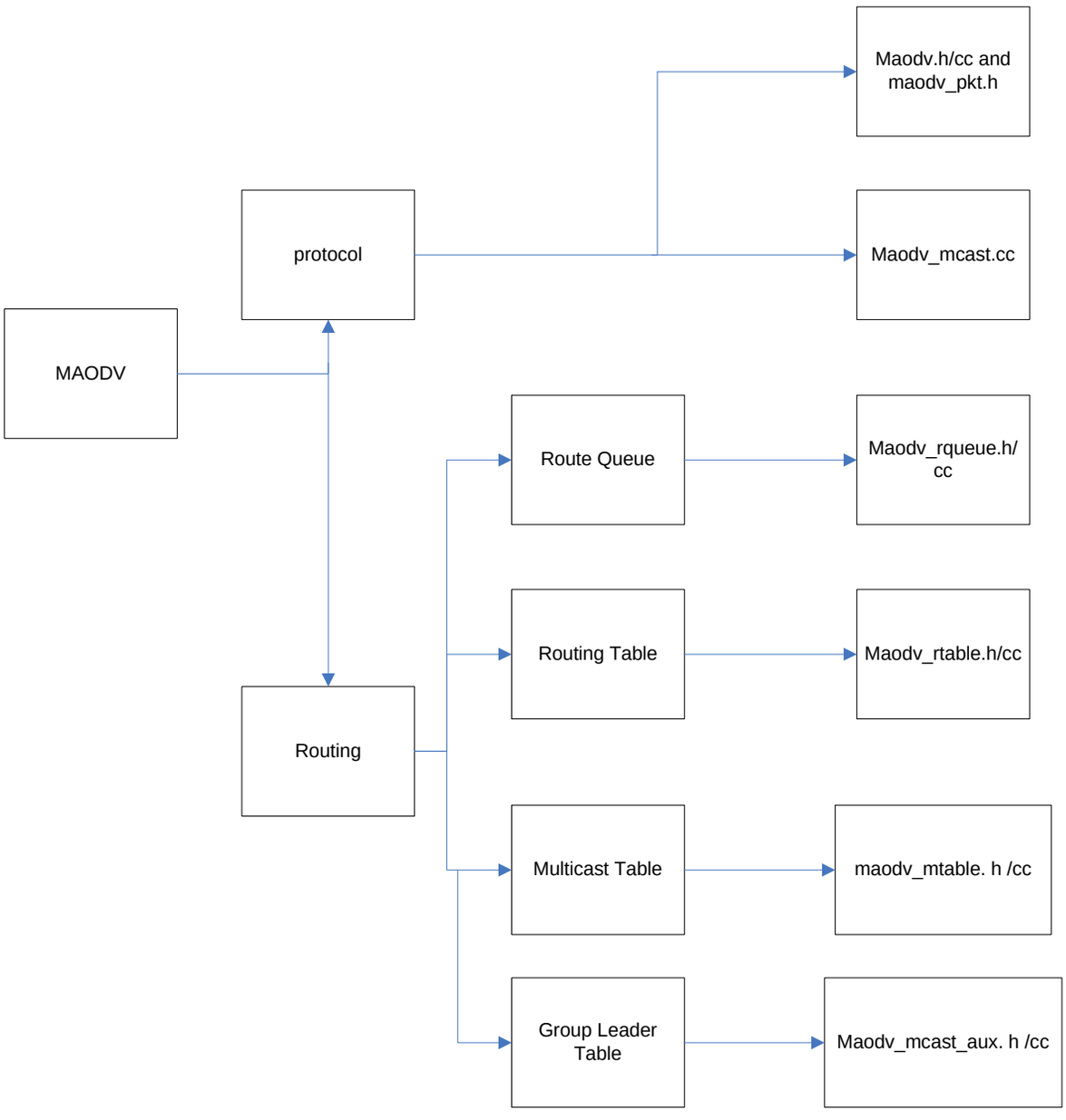


Figure 6.1 Block diagram of MAODV implementation in NS

In this MAODV consist of two main functions. First function covers the protocol implementation and Second function covers all the routing tables needed and their implementation details [11]. The description of the MAODV architecture is given as follows.

6.2 INTERNALS OF IMPLEMENTATION

Maodv_mcast_auc.cc/h

This file contains the implementation of the neighbor table and group leader table and all the functions needed for it.

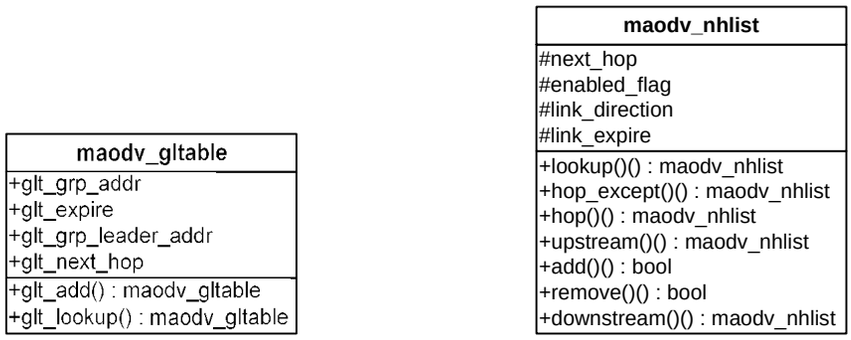


Figure 6.2 class for maodv_mcast_aux.h/cc

Maodv_gltable class contains all the details of the group leader table. Figure 6.2 shows the main attributes and function of the group leader class. Ii contains two main methods. First is add function which adds the nodes which are multicast group members and Second function lookup will search the table for particular entry of the node. It is needed when link breakage occur and Group Leader wants to delete the entry of the node

Maodv_nhlist class contains all the detail of the neighborhood table. Figure 6.2 shows the main attributes and functions of the neighborhood table. The lookup function will act same as in Group Leader table. Hop_except() and hop() function will check that sent packet will be sent to all the neighbor except the sender. Upstream() and downstream() function will return the upstream and donstream

nodes. Add() and Remove() function will simply add and remove the particular node.

Maodv_mtable.h/cc

This file contains all the details of multicast table (Figure 6.3). It contains mainly three functions. Lookup function will search for the entry of the particular node indicates that whether it is a multicast tree member or not. Add and delete function will simply add and delete a particular entry of a node. It is needed when timeout occur or when link breakage occur.

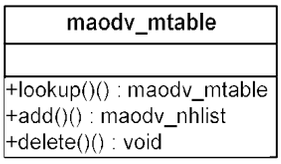


Figure 6.3 class for maodv_mtable

Maodv_rtable.h/cc

MAODV is based on AODV protocol so it uses its routing table. Figure 6.4 shows the main functions of the routing table, this routing table is used for unicast routing when receiver node is in the range of the sender node or when a node wants to join the group and it has the address of Group Leader, it can directly send unicast RREQ packet to Group Leader.

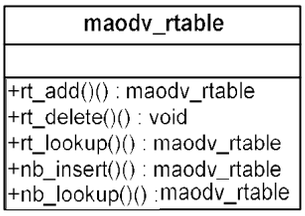


Figure 6.4 class for maodv_rtable

Rt_add() and Rt_delete() function will simply delete the entry for the node. And Rt_lookup function will work as in Group Leader table and multicast table for

finding a particular node. Nb_insert and Nb_lookup() node will insert and look for a node.

Maodv_rqueue.h/cc

This file contains all the details of routing queue. When packet arrives at the node it store in the queue and remove when node wants to forward the packet. Here for the simulation drop tail priority queue is used. After every 30seconds timeout occur and all the packets in the queue will be deleted. It also contains find method to find a particular destination packet from the queue. The maximum length of the queue is set to 64, means queue can hold up to 64 packets at a time.

Maodv_mcast.cc

This file contains all the functions needed for multicasting protocol. It contains details of the timers needed for multicasting such as prune timer, packet timer, RREPwait timer and grouphello timer. It also contains the implementation of all the functions regarding packets such as receiving MACT packet and forwarding it, receiving RERQ, RREP, GRPH, RI packets.

Maodv.h/cc

This is the main file, which contains details of the timers such as neighbor timer, broadcast timer, routecache timer and Localrepair timer. It also contains the implementation of some AODV functions (since MAODV is depended upon AODV protocol) and some multicasting functions such as sending the GRPH packet and other packets used for multicasting

6.3 Needed Changes

There are some files that are to be modified when one adds its new protocol in Network Simulator [11-13].

Packet.h

Since we add new protocol packet so it must be added in the common/paket.h file

Cmu-trace.h/cc

A trace object is used to write wanted information of a packet every time it is received, sent or dropped. For that we have to write format_maodv() function in the Cmu-trace.h file and its implementation is in cmu-trce.cc file.

Ns-packet.h

In this file we have to add the name of the new protocol in this case it is MADOV.

Ns-lib.tcl

We need a add procedure for creating a node. We have to create wireless node with MAODV as routing protocol. The procedure node from tcl file calls the create-wireless-ndoe procedure. So we have to write that procedure in ns.lib.tcl file.

Priqueue.cc

We need to tell the priqueue class that MAODV packets are routing packets and therefore treated as a high priority. For that we have to modify recv() function in the file.

Makefile

The last step after adding all the required function into the files, it should be compiled. The compilation of file is done by the "make" command. Before making the "make", the object files are to be added into the object list of the "Makefile". After adding and running the make command, if no errors comes than the protocol is successful and can be used for the simulation purpose.

7. RESULTS

7.1 SIMULATION SCENARIO

Simulation of protocol is done in NS2 [11, 13]. Following scenario is taken for the simulation

Area	: 1500 * 1500 m
No of Nodes	: 50
Simulation Duration	: 900 Sec
No of receivers	: 10, 20 , 30, 40, 50
No of senders	: 1, 5, 10
Transmission Range	: 250 m using Omni-directional antenna
Node movement Speed	: 1, 20
Traffic	: CBR 256bytes

The performance of the protocol is measured using two parameters [14].

PDR (Packet Delivery Ratio) = Total received Packet / (Total send Packets * no of receivers)

Latency is average Delay for data transfer from sender to receiver

Case 1: With Negligible Node Movement 1m/s

In this case original MAODV is simulated with negligible node movement with the different sender and receivers as shown in Table 7.1

7.1 MAODV with no node movement

	1 sender		5 sender		10 sender	
Receiver	PDR	Latency	PDR	Latency	PDR	Latency
10	0.9786	0.0287	0.9022	0.0271	0.8348	0.0360
20	0.9787	0.0282	0.8998	0.0315	0.8268	0.0372
30	0.9695	0.0320	0.8682	0.0342	0.8199	0.0389
40	0.9402	0.0269	0.8695	0.0325	0.8009	0.0378
50	0.9393	0.0275	0.8592	0.0329	0.7885	0.0355

Case 2: With Node Movement 20m/s

In this case MAODV is simulated with node high node movement (20m/s). The result of this simulation is shown in Table 7.2.

Table 7.2 MAODV With Node Movement 20m/s

	1 sender		5 sender		10 sender	
Receiver	PDR	Latency	PDR	Latency	PDR	Latency
10	0.9014	0.0276	0.8794	0.0265	0.7813	0.0344
20	0.9101	0.0279	0.8673	0.0318	0.7650	0.0360
30	0.9073	0.0299	0.8603	0.0300	0.7423	0.0382
40	0.9139	0.0262	0.8396	0.0319	0.7132	0.0372
50	0.8956	0.0269	0.8206	0.0300	0.7052	0.0322

Case 3: Node Movement 20m/s Including RI Packet And Making The GL Node Static

In this case MAODV with the improvement is simulated with node movement 20 m/s. The results are shown in Table 7.3.

Table 7.3 MAODV with RI and making GL node static (Node Movement 20m/s)

	1 sender		5 sender		10 sender	
Receiver	PDR	Latency	PDR	Latency	PDR	Latency
10	0.9400	0.0279	0.8809	0.0268	0.8287	0.0340
20	0.9350	0.0280	0.8765	0.0321	0.8184	0.0357
30	0.9252	0.0305	0.8725	0.0296	0.8052	0.0385
40	0.9201	0.0266	0.8656	0.0311	0.7920	0.0375
50	0.9088	0.0272	0.8581	0.0303	0.7745	0.0319

7.2 GRAPHS FOR PDR

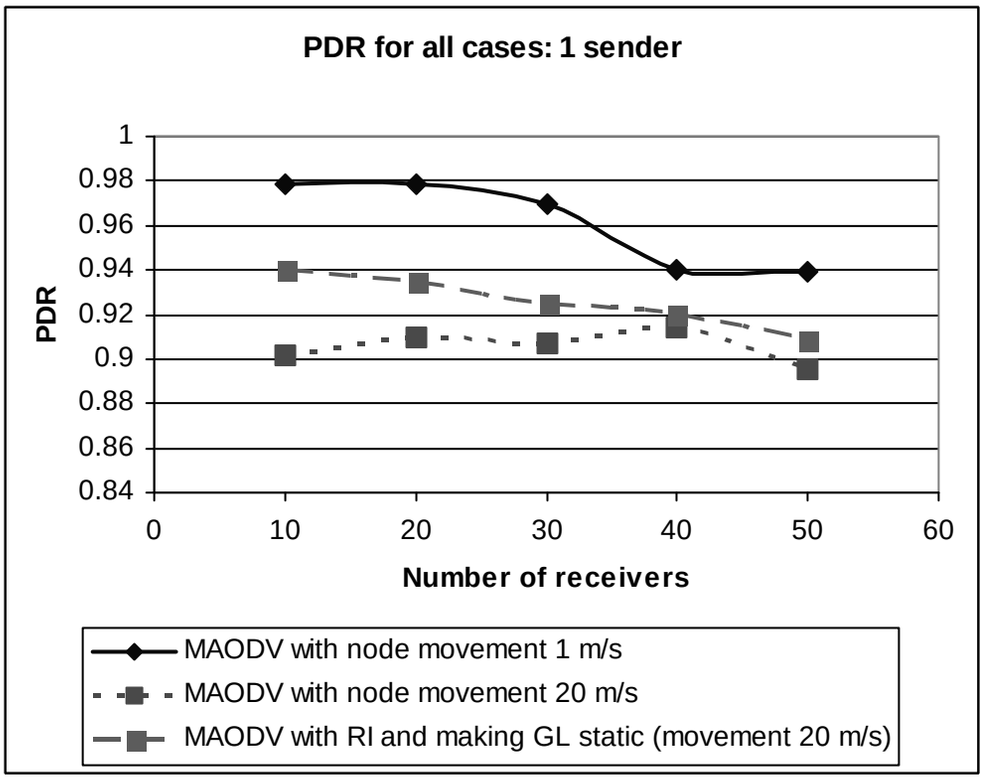


Figure 7.1: PDR for all the three cases: 1 senders

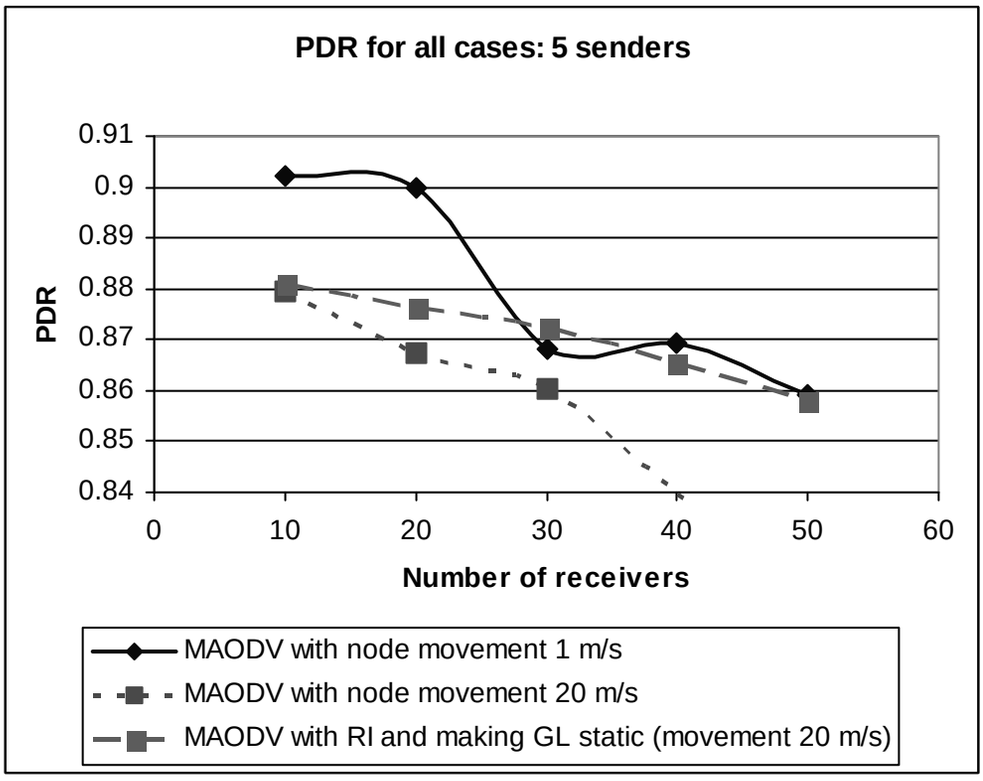


Figure 7.2: PDR for all the three cases: 5 senders

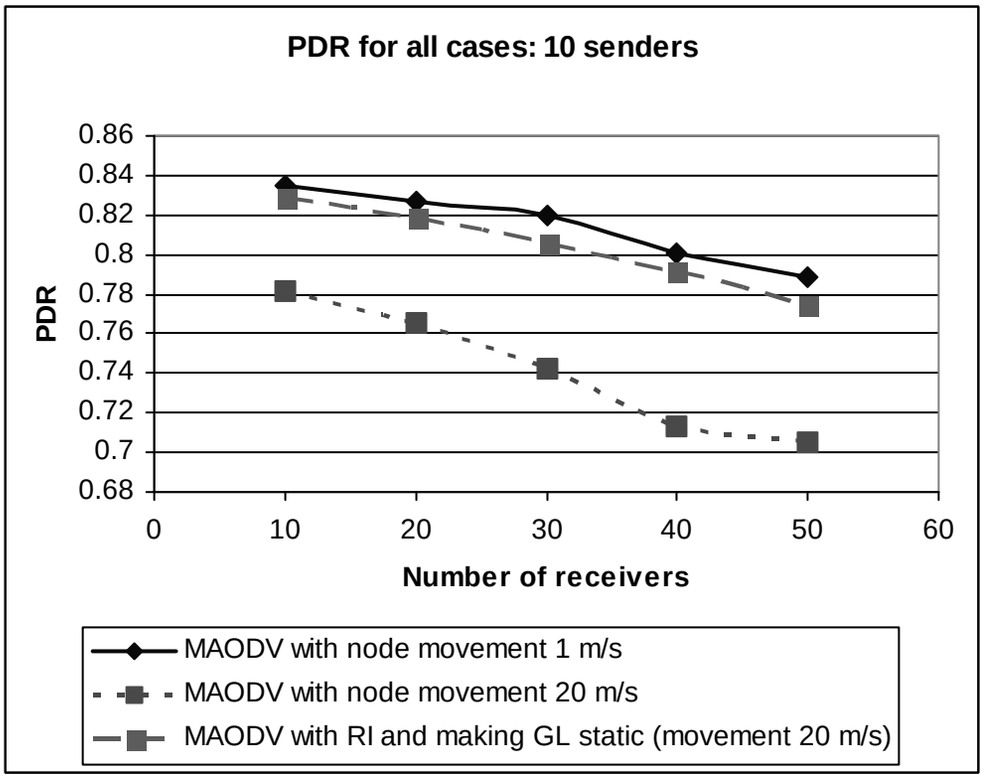


Figure 7.3: PDR for all cases: 10 senders

7.3 GRAPHS FOR THE LATENCY

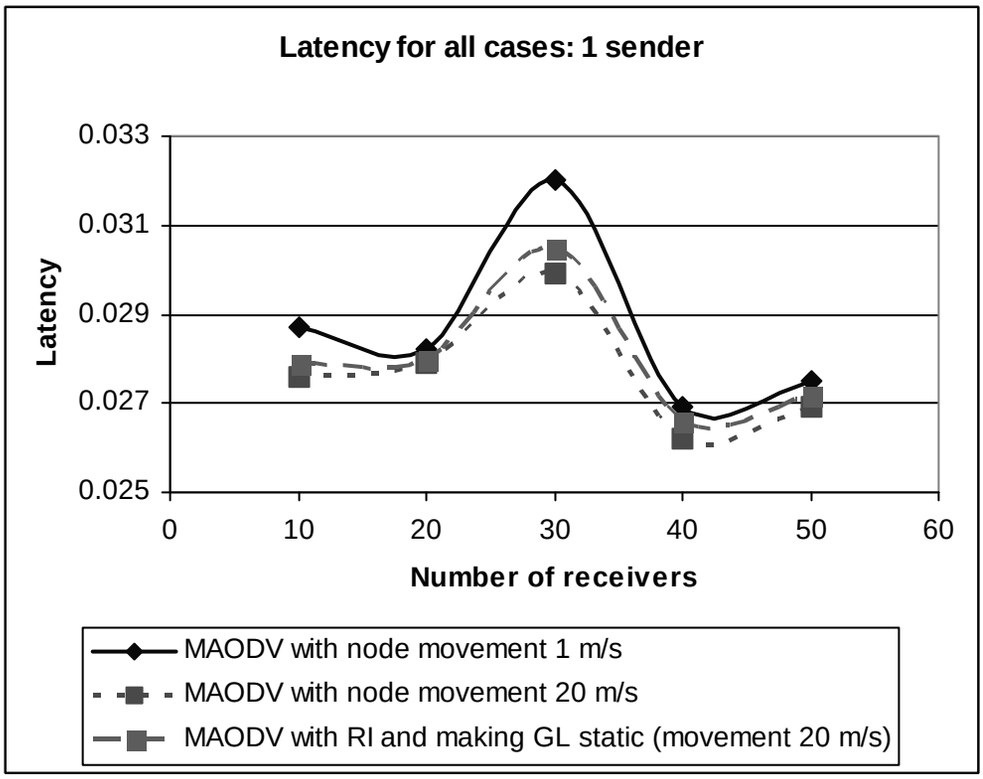


Figure 7.4 Latency for all the three cases: 1 sender

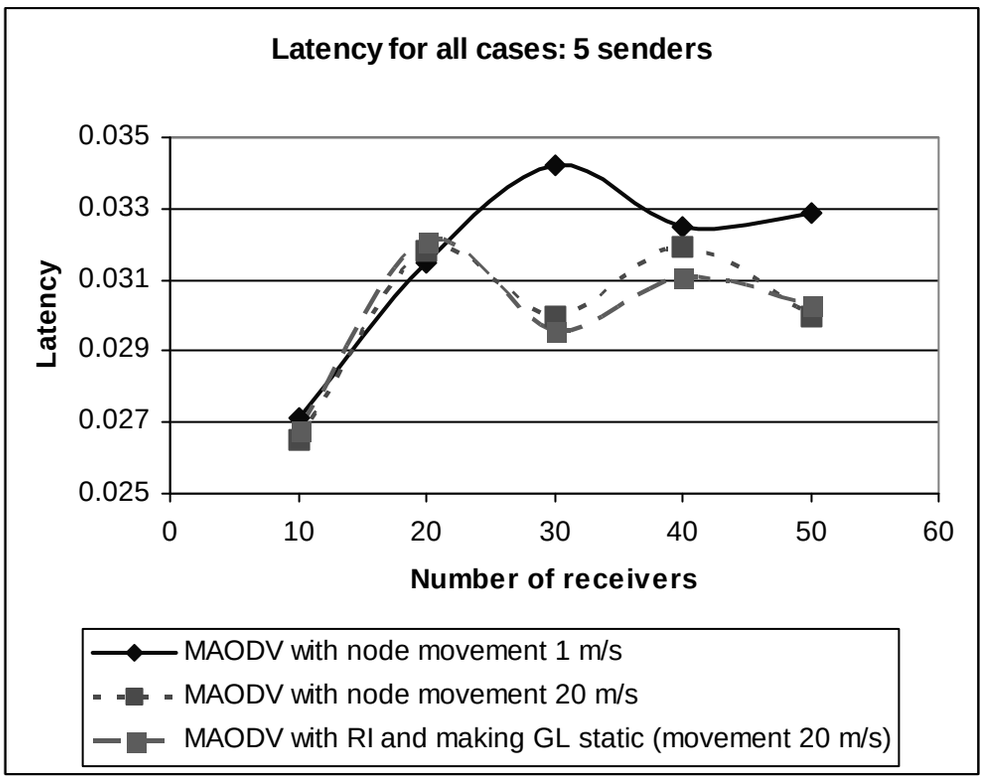


Figure 7.5 Latency for all cases: 5 senders

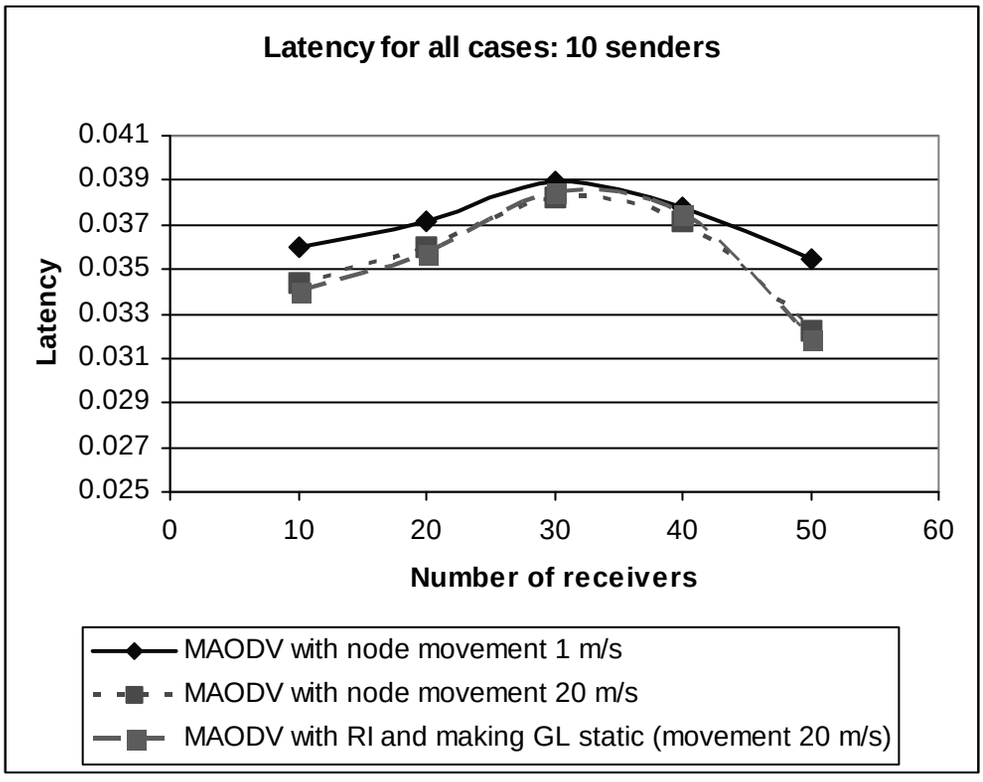


Figure 7.6 Latency for all cases: 10 senders

8. SUMMARY AND CONCLUSION

8.1 SUMMARY

The Multicast Ad hoc On-Demand Distance Vector (MAODV) protocol enables dynamic, self-starting, multihop routing between participating mobile nodes, which are ready to join or participate in a multicast group within an ad hoc network. One distinguishing feature of MAODV is its use of sequence numbers for multicast groups. Each multicast group has its own sequence number, which is initialized by the multicast group leader and incremented periodically.

The disadvantage turns up from its tree-based multicast topology, poor packet delivery under higher mobility. This protocol uses a shared tree approach and always carries along with it the risk of single point failure of the group leader, which might severely affect all multicast session in progress.

To overcome this disadvantage and improve its performance the work in this thesis focuses on adding new packet RI, which is transmitted periodically to check whether the receiver node is in the range of sender node or not, if it is in the range than sender directly sends packet to receiver rather than sending it through intermediate nodes. Here group leader node is also static so if any node wants to join the group, it can directly send RREQ packet to Group leader rather than broadcasting it. Through the results we can see that this approach improves the performance of a protocol under high mobility scenario.

8.2 CONCLUSION

All the graphs of PDR (Figure 7.1, 7.2, 7.3) shows that PDR of MAODV with this approach is in-between case 1(best possible results) and case 2(normal operation). It can also be seen that MAODV gives best possible result when node movement is negligible as 1m/s. When node movement is high as 20m/s the protocol performance starts degrading. But if we use RI packet and make Group Leader node static it gives good result than case 2(normal operation), which is nearer to best possible result (case 1).

All three graphs of Latency (Figure 7.4, 7.5, 7.6) shows that Latency for MAODV simulation for case 3 (use of RI and making Group Leader node static) gives better results than case 2 (normal operation) and it also give better results than case 1 (normal operation) with 10 sender and 30 receivers. It also shows that there is not much verification in latency for all the cases; means latency remains almost same for all the cases.

So MAODV performance, under high mobility scenario, in terms of PDR and Latency is better when we use RI packet and make the group leader node static in the simulation.

8.3 FUTURE WORK

In this thesis we consider only two approaches of MAODV that is add a new RI packet and make the group leader node static in the simulation but one can also use prediction technique i.e. Predict the failure or breakage of the link, so that after predict timeout, node will search a different route. Here we only make group leader node static, but one can also make number of node static at some fix points in the scenario and measure the performance.

REFERENCES

1. C. Siva Ram Murthy and B.S. Manoj, "Ad hoc Wireless Networks: Architecture and Protocols", India: Pearson Education Asia pte. ltd., 2005.

2. C. Diot et al., "Multipoint Communication: A Survey Of Protocols, Functions and Mechanisms." *IEEE Journal*, vol.15, n.3. April, 1997.

3. S. Deering, "Host extensions for IP multicasting," RFC-1112, August 1989.

4. Carlos de Morais Cordeiro, Hrishikesh Gossain, and Dharma P. Agrawal, "Multicast over wireless mobile ad-hoc networks: present and future directions", *IEEE Journal*, January/February 2003.

5. Upkar Varshney, "Multicast over wireless networks", *Communication of the ACM*, Vol. 45, no. 12, Dec 2002.

6. Elizabeth M. Royer, Santa Barbara, Chai-Keong Toh, "A review of current routing protocols for ad-hoc mobile wireless networks", *IEEE personal communications*, April 1999.

7. Elizabeth M. Royer, Charles E. Perkins, "Multicast Ad-Hoc On Demand Distance Vector Routing (MAODV)", Internet draft, http://www.ietf.org/internet-draft/ draft-ietf-manet-maodv-00, July 2000.

8. Tadao Murata, "Petri nets: Properties, Analysis, and Applications", *Proceedings of the IEEE*, Vol 77, No. 4, April 1989.

9. Gerard berthelot and Richard Terrat, "Petri Nets theory for the correctness of the protocol", *IEEE transactions of communications*, vol. Com-30, No. 12, December 1982.

10. The VNIT project, "The NS manual" , http://www.isi.edu/nsnam/ns/ns-Documentation, September 2006.

11. Francisco J. Ros and Pedro M. Ruiz, “Implementating a new MANET unicast routing protocol in ad-hoc networks”, Dept. of information and communication engineering, University of Murcia, December 2004

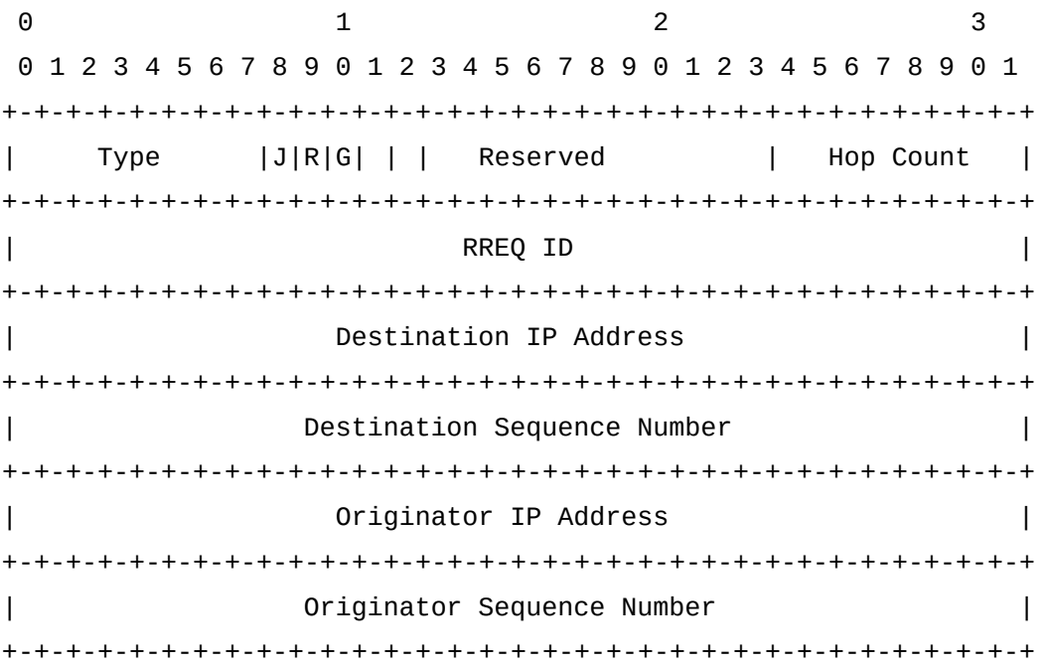
12. NS class hierarchy, http://www-sop.inria.fr/planete/software/ns-doc/ns-current/

13. NS by example, http://nile.wpi.edu/NS/

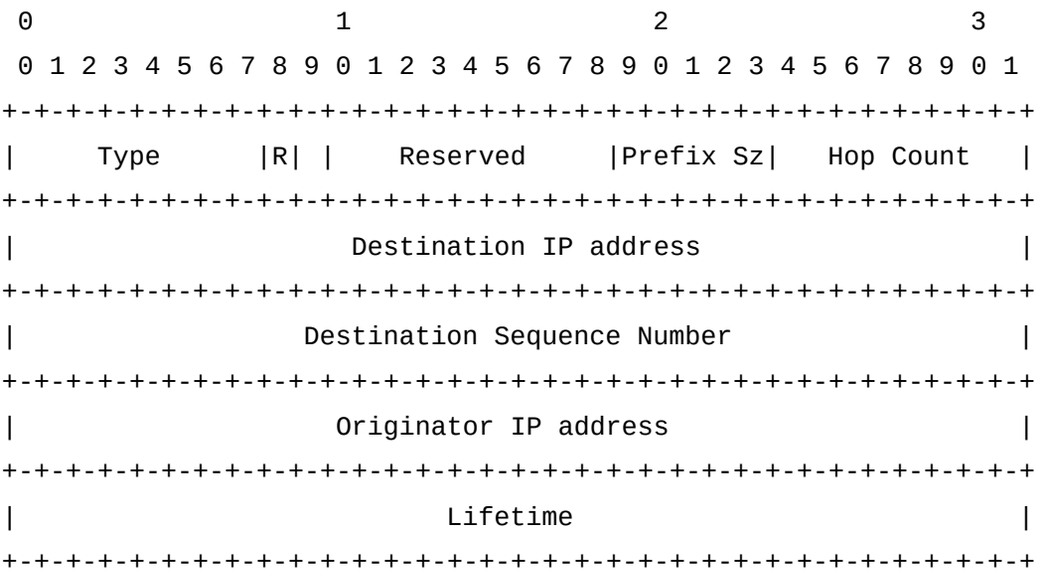
14. S. Carson and J. Macker, “Mobile Ad hoc networking (MANET) Routing Protocol Issues And Evaluation Consideration”, Internet draft, http://www.ietf.org/rfc/rfc2501.txt, July 2000.

APPENDIX A PACKET FORMATS

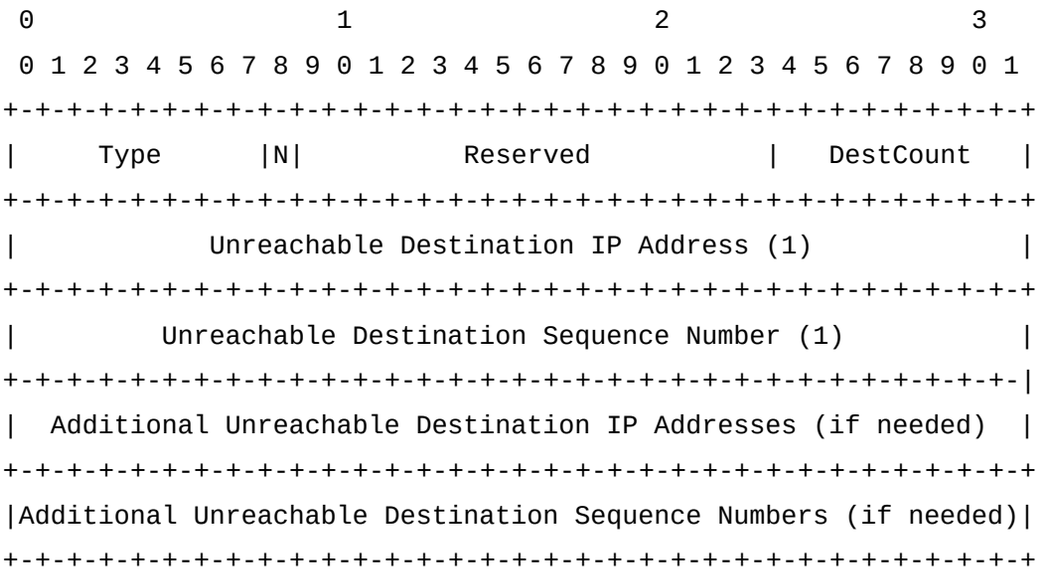
A.1 Route Request (RREQ) Message Format



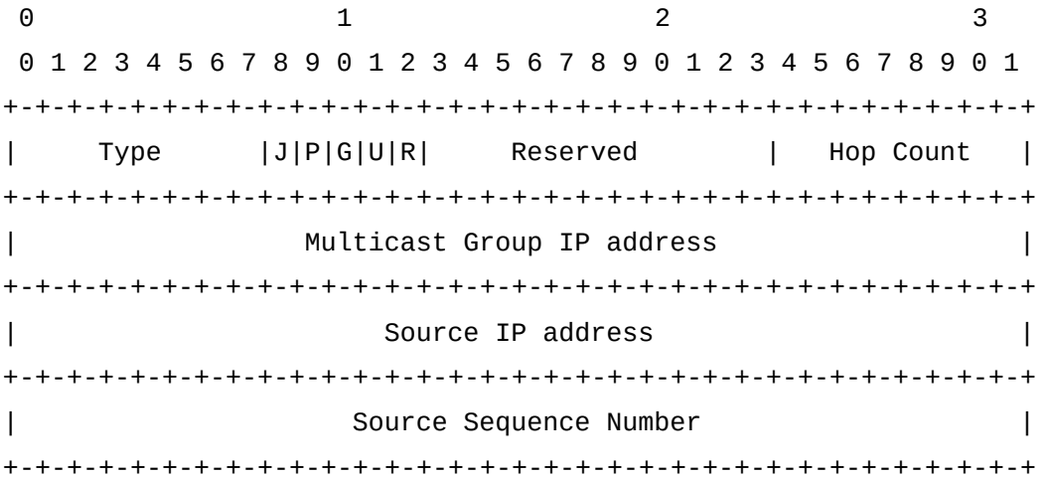
A.2 Route Reply (RREP) Message Format



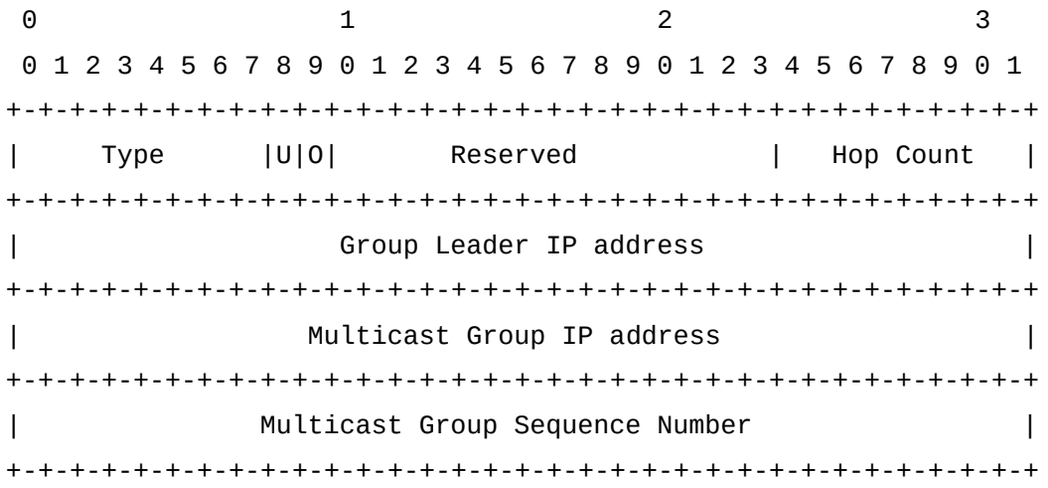
A.3 Route Error (RERR) Message Format



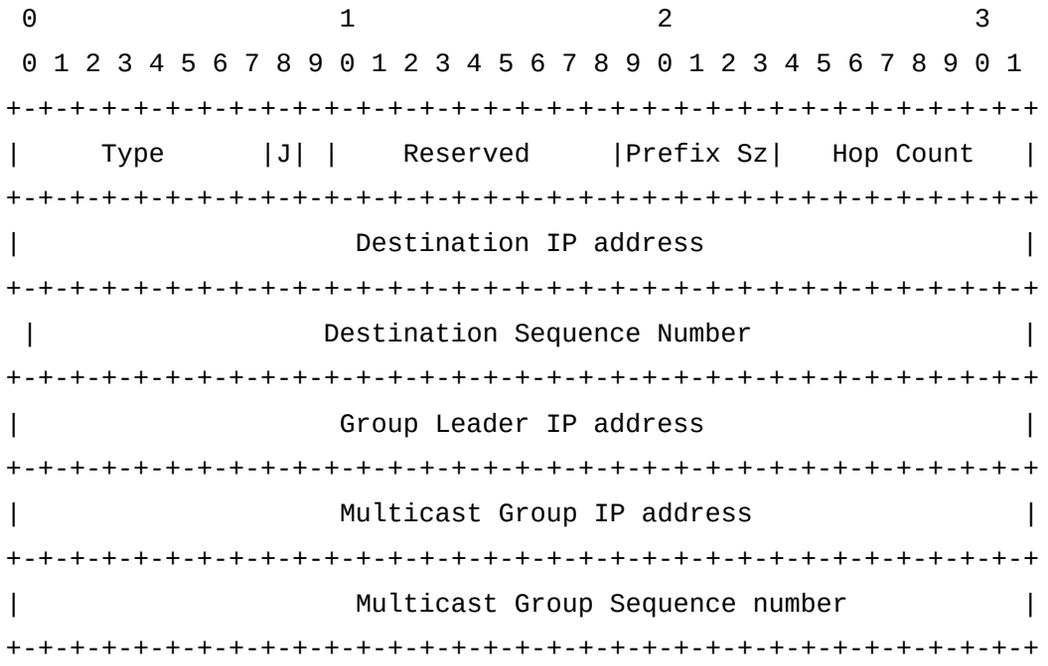
A.4 Multicast Activation (MACT) Message Format



A.5 Group Hello (GRPH) Message Format



A.6 Receiver Identification (RI) Message Format



APPENDIX B AWK SCRIPTS

When simulating any tcl script of NS2, two files are generated, one is NAM visualization file which shows the animation of network and other is trace file which contains the trace of every millisecond of simulation. The trace format of trace file is explained in chapter 5. This trace file contains only textual data, we cannot visualize form that data. So for calculating parameters likes jitter, delay, throughput, processing time etc awk scripts are used.

B.1 BASICS OF AWK

Awk is a full-featured text processing language with syntax reminiscent of **C**. While it possesses an extensive set of operators and capabilities. Awk breaks each line of input passed to it into fields. By default, a field is a string of consecutive characters delimited by white spaces, though there are options for changing this. Awk parses and operates on each separate field. This makes it ideal for handling structured text files -- especially tables -- data organized into consistent chunks, such as rows and columns.

- The structure of awk commands
    - Each awk command consists of a selector and/or an action; both may not be omitted in the same command. Braces surround the action.
    - selector [only] -- action is print
    - {action}[only] -- selector is every line
    - selector {action} -- perform action on each line where selector is true
    - Each action may have multiple statements separated from each other by semicolons or \n

- Line selection
    - A selector is either zero, one, or two selection criteria; in the latter case the criteria are separated by commas
    - A selection criterion may be either an RE or a Boolean expression (BE) which evaluates to true or false

- Commands which have no selection criteria are applied to each line of the input data set
- Commands which have one selection criterion are applied to every line which matches or makes true the criterion depending upon whether the criterion is an RE or a BE
- Commands which have two selection criteria are applied to the first line which matches the first criterion, the next line which matches the second criterion and all the lines between them.
- Unless a prior applied command has a next in it, every selector is tested against every line of the input data set.

- Processing
  - The BEGIN block(s) is(are) run (mawk's -v runs first)
  - Command line variables are assigned
  - For each line in the input data set
    - It is read and NR, NF, $I, etc. are set
    - For each command, its criteria are evaluated
    - If the criteria is true/matches the command is executed
  - After the input data set is exhausted, the END block(s) is(are) run
- Fields
  - Each record is separated into fields named **$1**, **$2**, etc
  - **$0** is the entire record
  - **NF** contains the number of fields in the current line
  - **FS** contains the field separator RE; it defaults to the white space RE, **/[<SPACE><TAB>]*/**
  - Fields may be accessed either by **$n** or by **$var** where **var** contains a value between **0** and **NF**

So by using awk script processes the trace file data and calculates different parameters of network.

- Delay = packet receive time at destination node – packet send time at source node.

- Packet Delivery Ratio (PDR) = Totla received packets/ Total sent packets * No of receiver