ENERGY AWARE MULTIPATH ON DEMAND DISTANCE VECTOR ROUTING IN MANETs

A DISSERTATION

Submitted in partial fulfillment of the requirements for the award of the degree
of
MASTER OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

By

MANE MANGESH RAMCHANDRA

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE -247 667 (INDIA)

JUNE, 2010

Candidates Declaration

I hereby declare that the work being presented in the dissertation report titled **"Energy Aware Multipath On Demand Distance Vector Routing in MANETs"** in partial fulfillment of the requirement for the award of the degree of **Master of Technology** in **Computer Science and Engineering**, submitted in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, is an authentic record of my own work carried out under the guidance of Dr. A. K. Sarje in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee. I have not submitted the matter embodied in this dissertation report for the award of any other degree.

Dated: 3|6|2010

Place: IIT, Roorkee

[illegible]

Mane Mangesh Ramchandra

Certificate

This is to certify that above statements made by the candidate are correct to the best of our knowledge and belief.

Dated:

Place: IIT, Roorkee

[illegible]
3|6|2010

Dr. A. K. Sarje,

Professor,

Department of Electronics and

Computer Engineering.

Acknowledgements

I express my deepest gratitude to Dr. A. K. Sarje, Professor, Department of Electronics and Computer Engineering, IIT Roorkee, for his valuable guidance, support and motivation in my work. I have deep sense of admiration for his inexhaustible enthusiasm and readiness to help me. The valuable discussions and suggestions with him have helped me a lot in supplementing my thoughts in the right direction for attaining the desired objective.

Special thanks to IIT Roorkee, for providing the necessary facilities to carry out this dissertation work.

I thank Mr. Sandeep Sood and Mr. Binay Kumar Pande, for their valuable support, which was helpful in making this dissertation work a success. I also thank Padmaja Kanase for the gift of friendship and for helping me in improving the quality of this report. Finally, I would like to thank my parents for their love, care, and support; friends, Rahul, Bharat, Sarika, Ragini, Sathish, Sanketh; my sister, Vidya and my brother Gurudev for their support.

List of Figures

List of Acronyms

AODV	Adhoc Ondemand Distance Vector
AOMDV	Adhoc Ondemand Multipath Distance Vector
ATHR	Adjust Threshold Message
CBR	Constant Bit Rate
IP	Internet protocol
JiST	Java in Simulation Time
JVM	Java Virtual Machine
LEAR-AODV	Local Energy Aware AODV
MAC	Media Access Control
MANET	Mobile Adhoc Network
MHRP	Minimum Hop Routing Protocol
RERR	Route Error Message
RREP	Route Reply Message
RREQ	Route Request Message
SWAN	Scalable Wireless Ad hoc Network Simulator
TTL	Time To Live
UDP	User Datagram Protocol

Abstract

In ad hoc networks, energy conservation is a very important design issue as most of the ad hoc network participants operate on battery. Since most routing protocols are flooding, conserving battery power in this process is essential. Most of the proposed routing protocols concentrate on finding and maintaining routes in the face of changing topology caused by either mobility or decreasing battery power. In this work we consider the routing problem in Mobile Ad hoc Networks (MANETs) with the goal of conserving battery power of nodes and reducing route discovery frequency.

We propose a routing algorithm that works better in static as well as dynamic conditions as compared to other routing protocols. Our approach is based on the formulation of multipath flow with consideration of remaining battery power of node. If remaining battery power of a node is small, then it will not take part in the route discovery process and will not forward packets on behalf of others. Hence battery power of the node will be preserved. The protocol is mutable for both static and dynamic ad hoc networks.

Performance comparison of our protocol with some well known protocol using JiST/SWAN simulator shows that our protocol is able to achieve a remarkable improvement in saving battery power, reducing end to end delay, and is also able to reduce routing overheads.

Table of Contents

Table of Contents

Chapter 1

Introduction

1.1. Introduction

The history of wireless networks started in the 1970s and the interest has been growing ever since. The tremendous growth of personal computers and the handy usage of mobile computers necessitate the need to share the information. The great popularity of Internet services make more people enjoy and depend on the networking applications. However, the Internet is not always available and reliable, and hence it cannot satisfy people's demand for communication at anytime and anywhere. A Mobile Ad hoc Network (MANET) [1] is a wireless network without any fixed infrastructure or centralized control. It contains mobile nodes that are connected dynamically in an arbitrary manner.

In MANETS, the nodes are the main components of the network. These nodes are mobile. They can move freely at any time, so the network structure changes dynamically. Each node behaves as a router; it takes part in discovery and maintains the routes to other nodes in the network. The main characteristics of the MANET are dynamic topology, bandwidth constrained, variable capacity links, energy constrained operation. The nodes can move freely at any time and can leave or join the network. The research challenges in MANET are related to routing, security, reliability, scalability, quality of services, internetworking, energy consumption and multimedia applications.

1.2. Motivation

Routing is one of the key issues in MANETs [1] due to their highly dynamic and distributed nature. In particular, energy efficient and multipath routing can be the most important design criteria for MANETs, because of following reasons.

1. The components of MANETs [1] are mostly battery operated devices. So the battery lifetime is one of the central issues. As each node acts as both host and router of packets, the battery of the host runs down very quickly if high traffic is routed through it. This leads to non-functioning of the node. This power failure of a mobile node not only affects the node itself but also its ability to forward packets on behalf of others and thus the overall network lifetime.

2. Traditional approaches for routing in mobile ad hoc networks adopt one single active path between source and destination nodes. The approach can be typically proactive or reactive [2]. But in dynamic conditions, i.e., when nodes are highly mobile, route breakage occurs very often. Multipath on-demand protocols [4] overcome this inefficiency, by allowing nodes to discover multiple disjoint routes between any source and destination nodes.

So development of multipath and energy efficient routing protocols is a key issue in supporting multi-hop communication. For this reason, a number of researchers have focused on the design of communication protocols that preserve energy and prevent network failures for as long as possible.

1.3. Problem statement

In this dissertation work we investigate the routing problem in Mobile Ad hoc Networks (MANETs) with the goal of preserving battery power of nodes and reducing route discovery frequency. We propose a routing algorithm that works better in static as well as dynamic conditions as compared to other routing protocols.
The main objective of present work can be stated as – "Development of an Energy aware multipath on demand distance vector routing in Mobile Ad Hoc Networks".

1.4. Organization of report

Chapter 2 discusses the routing strategies in MANETs, issues regarding energy aware and multipath routing. It also discusses three routing protocols AODV (Ad hoc On demand Distance Vector Routing), AOMDV (Ad hoc On demand Multipath Distance Vector routing), and LEAR-AODV (Local Energy Aware Routing) on which our protocol is based.

Chapter 3 introduces our routing algorithm. In this chapter we discuss design and working of our routing protocol.

Chapter 4 discusses the tools used for simulation of our routing protocol.

Chapter 5 discusses simulation setup on which experiments are carried out and the corresponding results obtained.

Finally chapter 6 concludes the dissertation work.

Background Studies

Chapter 2

Routing is the process of selecting paths in a network along which to send network traffic. Routing is one of the primary functions of MANETs which each node has to perform in order to enable connections between nodes that are not directly within each other's send range. Several routing protocols [2] have been proposed for mobile ad hoc networks. In our work we focus on energy aware and multipath routing. LEAR-AODV (Local Energy Aware Routing) [11] is one of the energy aware routing protocols, which aims to balance energy consumption among all participating nodes in an ad hoc network, and AOMDV(Ad hoc On demand Multipath Distance Vector) [8] is well known multipath routing protocol in MANETs. Both AMODV and LEAR-AODV are the extensions of the popular wireless routing protocol AODV (Ad hoc on demand distance vector) [4]. So in this chapter we discuss classification of routing protocols and some issues regarding multipath and energy aware routing and we briefly illustrates some of the key features of AODV, AMODV, and LEAR-AODV and shortcomings of these protocols to provide sufficient background for our proposal.

2.1 Classification of routing protocols in MANETs

There are different criteria for designing and classifying routing protocols for wireless ad hoc networks. For example, what routing information is exchanged; when and how the routing information is exchanged, when and how routes are computed etc. Some of the criteria discussed below.

2.1.1. Proactive vs. Reactive Routing

Proactive Schemes determine the routes to various nodes in the network in advance, so that the route is already present whenever needed. Route discovery overheads are large in such

schemes as one has to discover all the routes. They consume bandwidth to keep routes up-to-date. Packet forwarding is faster in these schemes as the route is already present.
Reactive Schemes determine the route when needed. Therefore they have smaller Route Discovery overheads. They employ a flooding (global search) mechanism. A node trying to transmit a packet may have to wait for route discovery. Examples of such schemes are Dynamic Source Routing, Ad-Hoc On Demand Distance Vector Routing (AODV) [4].

2.1.2. Table driven vs. Source Initiated Routing

In Table Driven Routing protocols, up-to-date routing information from each node to every other node in the network is maintained on each node of the network. The changes in network topology are then propagated in the entire network by means of updates. The routing protocols classified under Source Initiated On-Demand Routing, create routes only when desired by the source node. When a node requires a route to a certain destination, it initiates what is called as the route discovery process. This process basically comprises of packets with a description of the destination (address information of the destination etc.) being forwarded from one hop to the next. Any node receiving such a request looks into its available routing table to find if it has a route to the described destination. If a route to the destination is present, the node returns this route to the source and the process ends else the request packet is forwarded to its neighbors continuing the route search process. Once a route is found, it is temporarily maintained in some form (typically the routing table) and then subsequently removed after either a timeout, or if the destination node leaves the network etc.

2.1.3. Single path vs. multiple path Routing

There are several criteria for comparing single-path routing and multi-path routing in ad hoc networks. First, the overhead of route discovery in multi-path routing is much more than that of single-path routing. On the other hand, the frequency of route discovery is much less in a network which uses multi-path routing, since the system can still operate even if one or a few of the multiple paths between a source and a destination fail. Second, it is commonly believed that using multipath routing results in a higher throughput. The reason is that all nodes are assumed to have (and limited) capacity (bandwidth and processing power). Since multi-path routing distributes the load better, the overall throughput would be higher.

2.2. Energy Aware Routing

The first generation of routing protocols in ad hoc networks is essentially Minimum Hop Routing Protocols (MHRP) that do not consider energy efficiency as the main goal. While energy conservation becomes a major concern for the ad hoc network, many energy-aware routing algorithms have been proposed in recent years [21,22,23].
Singh et al. [9] propose several metrics for energy-aware routing:

- ***Minimize Energy Consumed/Packet.*** In this way, the total energy consumption of this network is minimized. However, it may cause some nodes to drain energy out faster since it tends to route packet around areas of congestion in the network.
- ***Maximize Time to Network Partition.*** Given a network topology, there exists a minimal set of nodes, the removal of which will cause the network to partition. The routes between these two partitions must go through one of these critical nodes. A routing procedure therefore must divide traffic among these nodes to maximize the lifetime of the network.
- ***Minimize Variance in Node Power Levels.*** The intuition behind this metric is that all nodes in the ad hoc network are of equal importance, and no node must be penalized more than any other nodes. This metric ensures that all the nodes in the network remain up and running together.
- ***Minimize Cost/Packet.*** In order to maximize the lifetime of all nodes in the network, metrics other than energy consumed/packet need to be used. The paths selected when using these metrics should be such that nodes with depleted energy reserves do not lie on many paths.
- ***Minimize Maximum Node Cost.*** This metric ensures that node failure is delayed. Unfortunately, there is no way to implement this metric directly in a routing protocol. However, minimizing the cost/node does significantly reduces the maximum node cost in the networks.

2.3 Multipath routing

The routing is the most active research field in the MANET. The routing protocols designed for wired networks are not suitable for wireless networks due to the node mobility issues in wireless networks. Due to node mobility, node failures, and the dynamic characteristics of the radio channel, links in a route may become temporarily unavailable and making the route invalid. The overhead of finding alternative routes may be high and extra delay in packet delivery may be introduced. The multipath routing addresses this problem by providing more than one route to a destination node. Multipath routing appears to be a promising technique for ad hoc routing protocols. Providing multiple routes is beneficial in network communications, particularly in MANETs, where routes become obsolete frequently because of mobility and poor wireless link quality. The source and intermediate nodes can use these routes as primary and backup routes. Alternatively, traffic can be distributed among multiple routes to enhance transmission reliability, provide load balancing, and secure data transmission. The multipath routing effectively reduces the frequency of route discovery therefore the latency for discovering another route is reduced when currently used route is broken. Multiple paths can be useful in improving the effective bandwidth of communication, responding to congestion and heavy traffic, and increasing delivery reliability.

2.4 Overview of AODV routing protocol

The Ad hoc On Demand Distance Vector Routing (AODV) [4] makes use of sequence numbers to supersede stale cached routes and to prevent looping. In AODV discovered route is stored locally at all intermediate nodes on the route. The route discovery process is initiated whenever a traffic source needs a route to a destination. Route discovery typically involves a network-wide flood of Route Request (RREQ) packets targeting the destination and waiting for a Route Reply (RREP). An intermediate node receiving an RREQ packet first sets up a reverse path to the source using the previous hop of the RREQ as the next hop on the reverse path. If a valid route to the destination is available, then the intermediate node generates an RREP; otherwise, the RREQ is rebroadcast. Duplicate copies of the RREQ packet received at any node are discarded. When the destination receives an RREQ, it also generates an RREP. The RREP is routed back to the source via the reverse path. As the RREP proceeds toward the source, a forward path to the destination is established.

Route maintenance is done using route error (RERR) packets. When a link failure is detected, an RERR is sent back via separately maintained predecessor links to all the sources using that failed link. Routes are erased by the RERR along its way. When a traffic source receives an RERR, it initiates a new route discovery if the route is still needed. Unused routes in the routing table are expired using a timer-based technique.

3.2 Overview of AOMDV routing protocol

AOMDV [8] is one of the most popular on-demand multipath protocols. It is an extension of a single-path routing scheme AODV, and it allows computation of multiple loop-free and link-disjoint paths between any source and destination nodes. In AOMDV, different instances of RREQs are not discarded by intermediate nodes as it in the AODV, because they may provide information about potential alternate reverse paths. If a new RREQ instance preserves the loop free condition and comes from a different last-hop node, then a new reverse route towards the source node is added in the routing table of intermediate nodes between source and destination. If one or more valid paths are available at a node, it generates RREP packet and forwards it back to source along the reverse path. If possible, the intermediate node includes in the new RREP a forward path that was not used in any previous RREP, for this RREQ. The intermediate node re-broadcasts the new RREQs to neighbor nodes. When the destination receives more RREQ instances, in order to get multiple link-disjoint routes, it replies with multiple RREP messages. A route with minimum hop count is selected for communication by nodes. AOMDV uses modified routing table as shown in Figure 2.2. It shows the structure of routing table entries for AODV and AOMDV. In AOMDV *Advertised_hopcount* replaces *Hop_count* in AODV which is maximum *Hop_count* of multiple paths for destination available at any particular node. A *Route_list* replaces *Next_hop*, all other fields are same as that in AODV.

Destination
Sequence_number
Hop _count
Next_hop
Expiration_ time

Figure 2.1.Structures of routing table entries of AODV

Destination
Sequence _number
Advertised_hopcount
Route_list *{(Next_hop1,Hopcount),(Next_hop2,Hopcount2), (Next_hop3,Hopcount3),.... }*
Expiration_time

Figure 2.2. Structures of routing table entries of AOMDV.

3.3 Overview of LEAR-AODV routing protocol

3.3.1. Route discovery in LEAR-AODV

In AODV, a mobile node has no choice and must forward packets for other nodes. But in LEAR-AODV, each node determines whether or not to accept and forward the RREQ message depending on its remaining battery power (*En*). When it is lower than a threshold value (*Th)* the RREQ is dropped (*En* < *Th*); otherwise, the message is forwarded. The destination will receive a route request message only when all intermediate nodes along the route have enough battery levels.

3.3.2. Route maintenance

Route maintenance is needed either when the connections between some nodes on the path are lost due to node mobility, or when the energy resources of some nodes on the path are depleting too quickly. In LEAR AODV when first case occurs a new RREQ is sent out, and the entry in the route table corresponding to the nodes that have moved out of range are removed. In the second case, the node sends an RERR back to the source even when the condition *(En<=Th)* is satisfied. This route error message forces the source to initiate route discovery again. This is a local decision because it is dependent only on the remaining battery capacity of the current node. However, if this decision is made for every possible route, the source will not receive an RREP message even if a route exists between the source and the destination.

To avoid this situation, the source will resend another RREQ message with an increased sequence number. When an intermediate node receives this new request, it lowers its *Th* by factor *q* to allow the packet forwarding to continue. LEAR-AODV uses a new control message, *ADJUST_Thr* for requesting to adjust power threshold of a node.

3.4. Shortcomings of AOMDV and LEAR-AODV

The components of MANETs are mostly battery operated devices and these devices might be highly mobile. So routing algorithms should consider these aspects while routing. The problem with AOMDV [8] and LEAR-AODV are as follows.

1. In AOMDV high focus is on finding multiple paths and selecting route with minimum hop count. In this process intermediate nodes in selected path have to forward packets on behalf of other nodes, even though their battery power levels are very low which causes those nodes to run down shortly.

2. In LEAR-AODV [11] main focus is to conserve the battery power of a node. The selected route is highly energetic. All the nodes in selected route have high remaining battery capacity. But in the dynamic conditions if nodes are very mobile, route will not be active for long time. That results into frequent restarting of route discovery process. So these frequent route discovery attempts and latency in route discovery affects the performance of MANET.

3. Another problem in LEAR-AODV is in the route discovery process when node receives RREQ message it will first check the remaining battery power level and if remaining power is less than threshold value ($En<=Th$) it will send RERR message. This condition is applicable to destination node also. So destination node will send RERR message if its remaining battery power is less than threshold. So source node will receive RERR message even though the route is available to destination.

4. And there is no any provision to distinguish RERR messages i.e. when node receive RERR message it won't be able to identify the cause of error. And even if there is error

because of mobility, originator node will broadcast ATHR(Adjust Threshold message) message that increases the overhead on network.

In next chapter we propose an energy conservative routing protocol which can tackle all the above mentioned problems.

Proposed Energy Aware Multipath Routing Protocol

Chapter 3

In this chapter we propose an energy aware routing protocol which can tackle all the problems mentioned in previous chapter. Our routing algorithm is mainly based on AOMDV [8] and LEAR-AODV [11]. We have integrated the properties of AOMDV and LEAR-AODV to form more reliable and efficient communication link between nodes participating in network. Our protocol is designed primarily for highly dynamic ad hoc networks where link failures and route breaks occur frequently. Another benefit for nodes which are more energy hungry is that they can save their energy by not forwarding packets on behalf of other if their battery power level is low.

3.1 Overview

The messages used by EAMR (Energy aware Multipath Routing) are Route Requests (RREQs), Route Replies (RREPs), Route Errors (RERRs) and Adjust Threshold (ATHR). These messages are received via UDP (User Datagram Protocol), and normal IP header processing applies. So, for instance, the requesting node is expected to use its IP address as the Originator IP address for the messages. For broadcast messages, the IP limited broadcast address (255.255.255.255) is used. This means that such messages are not blindly forwarded. However, EAMR operation does require certain messages (e.g., RREQ, ATHR) to be disseminated widely, perhaps throughout the ad hoc network. The range of dissemination of such RREQs is indicated by the TTL in the IP header. Fragmentation is typically not required.

As long as the endpoints of a communication connection have valid routes to each other, EAMR does not play any role. When a route to a new destination is needed, the node broadcasts a RREQ to find a route to the destination. A route can be determined when the

RREQ reaches either the destination itself, or an intermediate node with a 'fresh enough' route to the destination. A 'fresh enough' route is a valid route entry for the destination whose associated sequence number is at least as great as that contained in the RREQ. The route is made available by unicasting a RREP back to the originator of the RREQ. Each node receiving the request caches a route back to the originator of the request, so that the RREP can be unicast from the destination along a path to that originator, or likewise from any intermediate node that is able to satisfy the request.

Nodes monitor the link status of next hops in active routes and its own battery power status. When a link break in an active route is detected or if remaining battery power is less than threshold, a RERR message is used to notify other nodes that the loss of that link has occurred or nodes remaining battery power is less than threshold and it could not forward RREQ message. The RERR message indicates those destinations (possibly subnets) which are no longer reachable because of broken link or node having less battery power. In order to enable this reporting mechanism, each node keeps a "precursor list", containing the IP address for each of its neighbours that are likely to use it as a next hop towards each destination. The information in the precursor lists is most easily acquired during the processing for generation of a RREP message, which by definition has to be sent to a node in a precursor list. If the RREP has a nonzero prefix length, then the originator of the RREQ which solicited the RREP information is included among the precursors for the subnet route (not specifically for the particular destination).

Route table information must be kept even for short-lived routes, such as are created to temporarily store reverse paths towards nodes originating RREQs. EAMR uses the following fields with each route table entry:

- Destination IP Address
- Destination Sequence Number
- Valid Destination Sequence Number flag
- Other state and routing flags (e.g., valid, invalid, repairable, being repaired)
- Network Interface
- Link list (Hop Count (number of hops needed to reach destination) and corresponding Next Hop)

- Advertised_hopcount (section 3.7)
- List of Precursors (section 3.5)
- Lifetime (expiration or deletion time of the route)

Managing the sequence number is crucial to avoiding routing loops, even when links break and a node is no longer reachable to supply its own information about its sequence number. A destination becomes unreachable when a link breaks or is deactivated. When these conditions occur, the route is invalidated by operations involving the sequence number and marking the route table entry state as invalid.

3.2. Applicability Statement

The EAMR routing protocol is designed for mobile ad hoc networks with populations of tens to thousands of mobile nodes. EAMR can handle low, moderate, and relatively high mobility rates, as well as a variety of data traffic levels. EAMR is designed for use in networks where the nodes can all trust each other, either by use of preconfigured keys, or because it is known that there are no malicious intruder nodes. EAMR has been designed to reduce route discovery frequency, reduce relative battery consumption, reduce the dissemination of control traffic and eliminate overhead on data traffic, in order to improve scalability and performance.

3.3. Protocol Design

The flow chart shown in Figure 3.1. summarizes the action of an EAMR node when processing an incoming message. Detailed working is described in subsequent sections.

3.4. Message Formats

3.4.1. Route Request (RREQ) Message Format

The format of the Route Request message is illustrated in Figure 3.2. which contains the following fields:

- Type: 1.
- J: Join flag; reserved for multicast
- R: Repair flag; reserved for multicast.

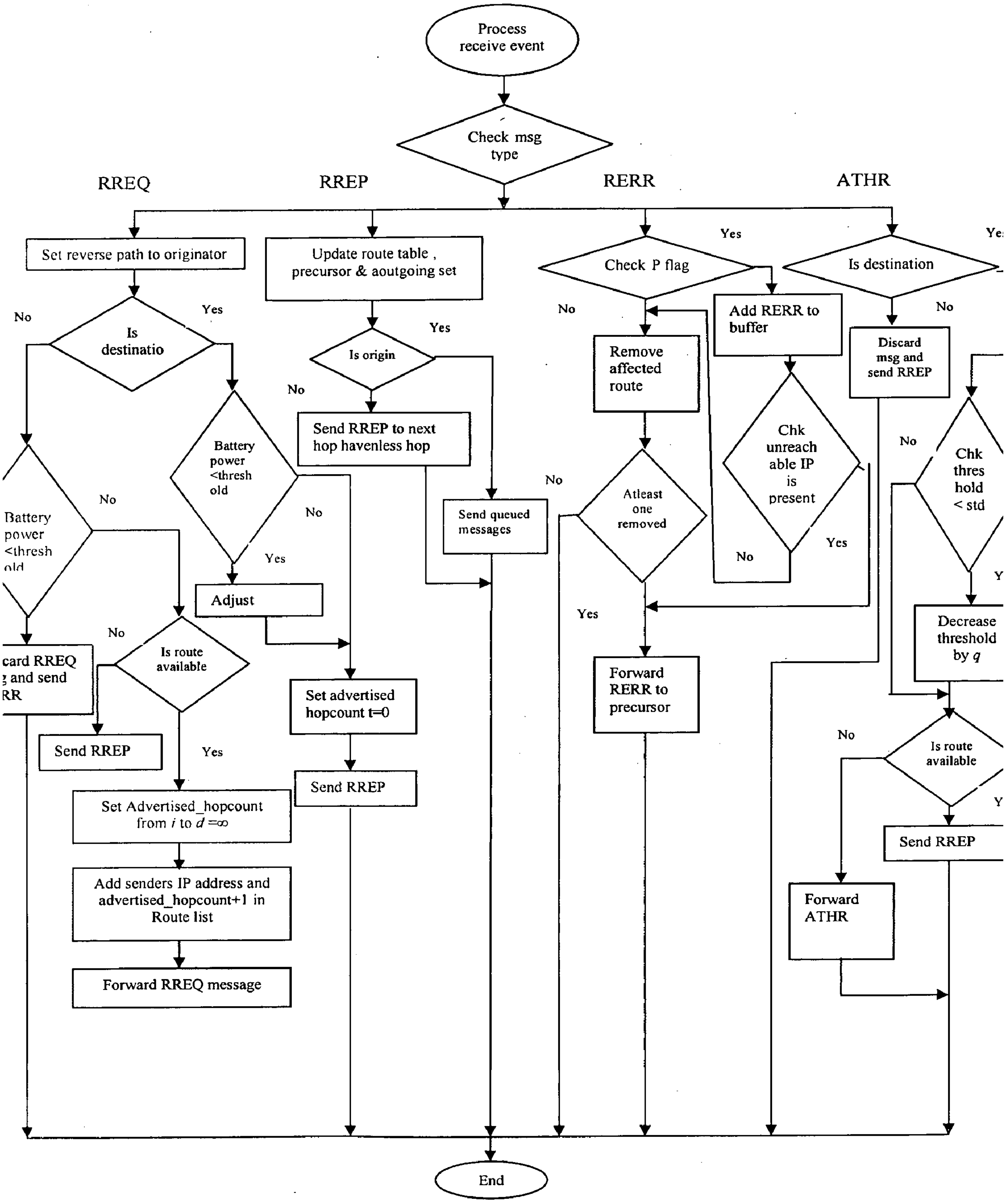


Figure 3.1. Flow chart of protocol

0 31

Type	J	R	G	D	U	Reserved	Hop count
RREQ ID							
Destinations IP Address							
Destination sequence Number							
Originator IP Address							
Lifetime							

Figure 3.2. Route request message format

- G: Gratuitous RREP flag; indicates whether a gratuitous RREP should be unicast to the node specified in the Destination IP Address field.
- D: Destination only flag; indicates only the destination may respond to this RREQ.
- U: Unknown sequence number; indicates the destination sequence number is unknown.
- Reserved: Sent as 0; ignored on reception.
- Hop Count: The number of hops from the Originator IP Address to the node handling the request.
- RREQ ID: A sequence number uniquely identifying the particular RREQ when taken in conjunction with the originating node's IP address.
- Destination IP Address: The IP address of the destination for which a route is desired.
- Destination Sequence Number: The latest sequence number received in the past by the originator for any route towards the destination.
- Originator IP Address: The IP address of the node which originated the Route Request.
- Originator Sequence Number: The current sequence number to be used in the route entry pointing towards the originator of the route request.

3.4.2. Route Reply (RREP) Message Format

0 31

Type	R	A	Reserved	Prefix	Hop count
Destinations IP Address					
Destination sequence Number					
Originator IP Address					
Lifetime					

Figure 3.3. Route reply message format

The format of the Route Reply message is illustrated above, and contains the following fields:

- Type: 2.
- R: Repair flag; used for multicast.
- A: Acknowledgment required; see sections 5.4 and 6.7.
- Reserved: Sent as 0; ignored on reception.

3.4.3. Route Error (RERR) Message Format

0 31

Type	N	P	Reserved	DestCount
Unreachable Destinations IP Address(1)				
Unreachable Destination sequence Number(1)				
Additional Unreachable Destination IP Addresses (if needed)				
Additional Unreachable Destination Sequence Numbers (if needed)				

Figure 3.4 Route error message format

The format of the Route Error message is illustrated above, and contains the following fields:

- Type: 3.
- N: No delete flag; set when a node has performed a local repair of a link, and upstream nodes should not delete the route.
- P: Power error flag; set when node has less battery power remaining
- Reserved: Sent as 0; ignored on reception.
- DestCount: The number of unreachable destinations included in the message; must

be at least 1.

- ➢ Unreachable Destination IP Address: The IP address of the destination that has become unreachable due to a link break.
- ➢ Unreachable Destination Sequence Number: The sequence number in the route table entry for the destination listed in the previous Unreachable Destination IP Address field.

The RERR message is sent whenever a node power is less than power threshold or link break causes one or more destinations to become unreachable from some of the node's neighbours.

3.4.4. Adjust threshold message format

0 31

Type	Reserved
RREQ ID	
Destination IP Address	
Destination sequence Number	
Originator IP Address	
Originator Sequence Number	

Figure 3.5Adjust threshold message format

The format of the Route Error message is illustrated above, and contains the following fields:

- ➢ Type: 4
- ➢ Reserved: Sent as 0;

Other fields are same as RREQ message.

3.5. Route Table Entries and Precursor Lists

When a node receives an ECMR control packet from a neighbour, or creates or updates a route for a particular destination or subnet, it checks its route table for an entry for the destination. In the event that there is no corresponding entry for that destination, an entry is created. The sequence number is either determined from the information contained in the control packet, or else the valid sequence number field is set to false. The route is only updated if the new sequence number is either

1. Higher than the destination sequence number in the route table, or

2. The sequence numbers are equal, but the Advertised hop count (of the new information) plus One, is smaller than the existing Advertised hop count in the routing table, or

3. The sequence number is unknown.

The Lifetime field of the routing table entry is either determined from the control packet, or it is initialized to ACTIVE_ROUTE_TIMEOUT. This route may now be used to send any queued data packets and fulfils any outstanding route requests. Each time a route is used to forward a data packet, its Active Route Lifetime field of the source, destination and the next hop on the path to the destination is updated to be no less than the current time plus ACTIVE_ROUTE_TIMEOUT. Since the route between each originator and destination pair is expected to be symmetric, the Active Route Lifetime for the previous hop, along the reverse path back to the IP source, is also updated to be no less than the current time plus ACTIVE_ROUTE_TIMEOUT. The lifetime for an Active Route is updated each time the route is used regardless of whether the destination is a single node or a subnet.

For each valid route maintained by a node as a routing table entry, the node also maintains a list of precursors that may be forwarding packets on this route. These precursors will receive notifications from the node in the event of detection of the loss of the next hop link. The list of precursors in a routing table entry contains those neighbouring nodes to which a route reply was generated or forwarded.

3.6. Generating Route Requests

A node disseminates a RREQ when it determines that it needs a route to a destination and does not have one available. This can happen if the destination is previously unknown to the node, or if a previously valid route to the destination expire or is marked as invalid. The Destination Sequence Number field in the RREQ message is the last known destination sequence number for this destination and is copied from the Destination Sequence Number field in the routing table. If no sequence number is known, the unknown sequence number flag MUST be set. The Originator Sequence Number in the RREQ message is the node's own sequence number, which is incremented prior to insertion in a RREQ. The RREQ ID field is incremented by one from the last RREQ ID used by the current node. Each node maintains only one RREQ ID. The Advertised_ hopcount field is set to zero.

Before broadcasting the RREQ, the originating node buffers the RREQ ID and the Originator IP address (its own address) of the RREQ for PATH_DISCOVERY_TIME. In this way, when the node receives the packet again from its neighbours, it will not reprocess and re-forward the packet. An originating node often expects to have bidirectional communications with a destination node. In such cases, it is not sufficient for the originating node to have a route to the destination node; the destination must also have a route back to the originating node. In order for this to happen as efficiently as possible, any generation of a RREP by an intermediate node for delivery to the originating node should be accompanied by some action that notifies the destination about a route back to the originating node. The originating node selects this mode of operation in the intermediate nodes by setting the 'G' flag. See section 3.8.1. for details about actions taken by the intermediate node in response to a RREQ with the 'G' flag set.

A node will not originate more than RREQ_RATELIMIT RREQ messages per second. After broadcasting a RREQ, a node waits for a RREP (or other control message with current information regarding a route to the appropriate destination). If a route is not received within NET_TRAVERSAL_TIME milliseconds, the node may try again to discover a route by broadcasting another RREQ, up to a maximum of RREQ_RETRIES times at the maximum TTL value. Each new attempt must increment and update the RREQ ID. For each attempt, the TTL field of the IP header is set according to the mechanism specified in Appendix A, in order to enable control over how far the RREQ is disseminated for the each retry.

Data packets waiting for a route (i.e., waiting for a RREP after a RREQ has been sent) is buffered. The buffering is "first-in, first-out" (FIFO). If a route discovery has been attempted RREQ_RETRIES times at the maximum TTL without receiving any RREP, and if node has received any RERR message with P flag is on then node will send ATHR message otherwise all data packets destined for the corresponding destination will dropped from the buffer and a Destination Unreachable message will delivered to the application.

To reduce congestion in a network, repeated attempts by a source node at route discovery for a single destination must utilize a binary exponential backoff. The first time a source node broadcasts a RREQ, it waits NET_TRAVERSAL_TIME milliseconds for the reception of a

RREP. If a RREP is not received within that time, the source node sends a new RREQ. When calculating the time to wait for the RREP after sending the second RREQ, the source node MUST use a binary exponential backoff. Hence, the waiting time for the RREP corresponding to the second RREQ is 2 * NET_TRAVERSAL_TIME milliseconds. If a RREP is not received within this time period, another RREQ may be sent, up to RREQ_RETRIES additional attempts after the first RREQ. For each additional attempt, the waiting time for the RREP is multiplied by 2, so that the time conforms to a binary exponential backoff.

3.7. Processing and Forwarding Route Requests

When a node receives a RREQ, it first creates or updates a route to the previous hop then checks to determine whether remaining battery power is less then threshold value, if node has less battery power than threshold then node will simply discard that message and send RERR message to originator with P flag set to on. If not then it checks whether it has received a RREQ with the same Originator IP Address and RREQ ID within at least the last PATH_DISCOVERY_TIME. If such a RREQ has been received, then node will not discards the newly received RREQ because each route advertisement arriving at a node during EAMR route discovery potentially defines an alternate path to the source or the destination. For example, each copy of the RREQ packet arriving at a node defines an alternate path back to the source. However, accepting all such copies naively to construct routes will lead to routing loops.

In order to eliminate any possibility of loops, we maintain a similar invariant as in the single path case. The major difference, however, is that we accept and maintain multiple next-hop routes as obtained by multiple route advertisements, but we do this as long as the invariant is satisfied. One trouble is that different routes to the same destination now may have different hopcounts. Therefore, a node must be consistent regarding which one of these multiple hopcounts is advertised to others. It should not advertise different hopcounts to different neighbors with the same destination sequence number.

We used invariant based on notion of "advertised hopcount". The *Advertised hopcount* of a node *i* for a destination *d* represents the "maximum" hopcount of the multiple paths for *d* available at *i*. "Maximum" hopcount is considered, as then the advertised hopcount can never change for the same sequence number. The protocol only allows accepting alternate routes with lower hopcounts. This invariance is necessary to guarantee loop freedom. *Advertised_hopcount* is initialized each time the sequence number is updated. A node *i* updates its *Advertised_hopcount* for destination *d* as follows.

$$Advertised\_hopcount_i^d := \max_k = \{Hop\_count_k \mid (Next\_hop, Hop\_count) \in Route\_list_i^d\}$$

Route discovery rule for Energy Conservative Multipath Routing is shown in Figure.3.6. which is invoked whenever node *i* receives RREQ message to destination *d* from a neighbor *j* the variables $seqnum_i^d$, $Advertised\_hopcount_i^d$ and $Route\_list_i^d$ represent the sequence number, *Advertised_hopcount* and *Route_list* for destination *d* at node *i* respectively. In Figure.3.6. Line 1. represent energy conserving condition, that is when node is not a destination and it is having less remaining battery power then it will send RERR message and node will not take part in route discovery any more. Line 5, 14 and 15 ensure loop freedom.

```
1.  if ( ($PowerOfNode_i$ <= $Threshold$) and ($i \neq d$)) then
2.      send($RERR_j$);
3.      return ;
4.  else
5.      if($seqnum_i^d < seqnum_j^d$) then
6.          $seqnum_i^d = seqnum_j^d$;
7.          if($i \neq d$) then
8.              $Advertised\_hopcount_i^d = \infty$ ;
9               $Route\_list =$ NULL ;
10.             insert($j, Advertised\_hopcount_j^d + 1$) into ($Route + list_i^d$)
11.         else
12.             $Advertised\_hopcount_i^d = 0$;
13          endif
14      elseif ( $seqnum_i^d = seqnum_j^d$) and (($Advertised\_hopcount_i^d, i$) > ($Advertised\_hopcount_j^d, j$)) then
15              insert($j, Advertised\_hopcount_j^d + 1$) into $Route\_list_i^d$;
16      endif
17. endif
```

Figure 3.6. Route discovery rule.

Since participation in routing process is local decision of node there is no requirement in any change in RREQ message for power constrain.

Whenever a RREQ message is received, the Lifetime of the reverse route entry for the Originator IP address is set to be the maximum of (ExistingLifetime, MinimalLifetime), where

MinimalLifetime = (current time + 2*NET_TRAVERSAL_TIME - 2*HopCount*NODE_TRAVERSAL_TIME).

The current node can use the reverse route to forward data packets in the same way as for any other route in the routing table.

If a node does not generate a RREP, and if the incoming IP header has TTL larger than 1, the node updates and broadcasts the RREQ to address 255.255.255.255 on each of its configured interfaces. To update the RREQ, the TTL outgoing IP header is decreased by one. Lastly, the Destination Sequence number for the requested destination is set to the maximum of the corresponding value received in the RREQ message, and the destination sequence value currently maintained by the node for the requested destination. However, the forwarding node not modify its maintained value for the destination sequence number, even if the value received in the incoming RREQ is larger than the value currently maintained by the forwarding node.

Otherwise, if a node generates a RREP, then it discards the RREQ. Notice that, if intermediate nodes reply to every transmission of RREQs for a particular destination, it might turn out that the destination does not receive any of the discovery messages. In this situation, the destination does not learn of a route to the originating node from the RREQ messages. This could cause the destination to initiate a route discovery (for example, if the originator is attempting to establish a TCP session). In order that the destination learn of routes to the originating node, the originating node set the "gratuitous RREP" ('G') flag in the RREQ if for any reason the destination is likely to need a route to the originating node. If, in response to a RREQ with the 'G' flag set, an intermediate node returns a RREP, it MUST also unicast a gratuitous RREP to the destination node.

3.8. Generating Route Replies

A node generates a RREP if either:

1. It is itself the destination, or

2. It has an active route to the destination, the destination sequence number in the node's existing route table entry for the destination is valid and greater than or equal to the Destination Sequence Number of the RREQ (comparison using signed 32-bit arithmetic), and the "destination only" ('D') flag is NOT set.

When generating a RREP message, a node copies the Destination IP Address and the Originator Sequence Number from the RREQ message into the corresponding fields in the RREP message. Processing is slightly different, depending on whether the node is itself the requested destination, or instead if it is an intermediate node with a fresh enough route to the destination.

Once created, the RREP is unicast to the next hop toward the originator of the RREQ, as indicated by the route table entry for that originator. As the RREP is forwarded back towards the node which originated the RREQ message, the Hop Count field is incremented by one at each hop. Thus, when the RREP reaches the originator, the Hop Count represents the distance, in hops, of the destination from the originator.

3.8.1. Route Reply Generation by the Destination

If the generating node is the destination itself, it MUST increment its own sequence number by one if the sequence number in the RREQ packet is equal to that incremented value. Otherwise, the destination does not change its sequence number before generating the RREP message. The destination node places its (perhaps newly incremented) sequence number into the Destination Sequence Number field of the RREP, and enters the value zero in the Hop Count field of the RREP.

3.8.2. Route Reply Generation by an Intermediate Node

If the node generating the RREP is not the destination node, but instead is an intermediate hop along the path from the originator to the destination, it copies its known sequence number for the destination into the Destination Sequence Number field in the RREP message.

The intermediate node updates the forward route entry by placing the last hop node (from which it received the RREQ, as indicated by the source IP address field in the IP header) into the precursor list for the forward route entry i.e., the entry for the Destination IP Address. The intermediate node also updates its route table entry for the node originating the RREQ by placing the next hop towards the destination in the precursor list for the reverse route entry i.e., the entry for the Originator IP Address field of the RREQ message data.

3.8.3. Generating Gratuitous RREPs

If the RREQ has the 'G' flag set, and the intermediate node returns a RREP to the originating node, it unicasts a gratuitous RREP to the destination node.

The gratuitous RREP is then sent to the next hop along the path to the destination node, just as if the destination node had already issued a RREQ for the originating node and this RREP was produced in response to that (fictitious) RREQ. The RREP that is sent to the originator of the RREQ is the same whether or not the 'G' bit is set.

3.9. Receiving and Forwarding Route Replies

When a node receives a RREP message, it searches minimum hop count and next hop for a route to the previous hop from the list in route table entry. The node then increments the hopcount value in the RREP by one, to account for the new hop through the intermediate node. Call this incremented value the "New Hop Count". Then the forward route for this destination is created if it does not already exist. Otherwise, the node compares the Destination Sequence Number in the message with its own stored destination sequence number for the Destination IP Address in the RREP message.

3.10. Generating and processing Adjust threshold message

Node generate Adjust threshold message when a route discovery has been attempted RREQ_RETRIES times at the maximum TTL without receiving any RREP, and if node has received any RERR message with P flag is set. Flag P indicate that there might be route available to destination but node had less power than threshold.

Node broadcasts ATHR message to indicate neighboring node that there is no route available to destination so threshold value of nodes should be adjusted. When node receives ATHR message node will adjust their power threshold value by q factor which is same for every node.

Whenever node adjust their threshold value it checks whether it has valid route to the destination which is mentioned in ATHR message if that so then node will generate RREP message and send to originator. If there is no route available then node will simply adjust their power threshold value and forward ATHR message.

Node will just not blindly adjust their threshold value if there is no need to adjust value it will check if there is need of adjusting threshold value by comparing their threshold value to standard threshold value which is same to all the nodes.

After sending ATHR message, originator node will wait for NET_TRAVERSAL_TIME and then it will again send RREQ message and this time TTL value is set to maximum to avoid repeated sending RREQ message.

4,1.2. JiST Architecture

The JiST system architecture, depicted in Figure 4.1, consists of four distinct components: a compiler, a bytecode rewriter, a simulation kernel and a virtual machine. One writes JiST simulation programs in plain, unmodified Java and compiles them to bytecode using a regular Java language compiler. These compiled classes are then modified, via a bytecode-level rewriter, to run over a simulation kernel and to support the *simulation time* semantics described shortly. The simulation program, the rewriter and the JiST kernel are all written in pure Java. Thus, this entire process occurs within a standard, unmodified Java virtual machine (JVM). The benefits of this approach to simulator construction over traditional systems and languages approaches are numerous. Embedding the simulation semantics within the Java language allows us to reuse a large body of work, including the Java language itself, its standard libraries and existing compilers. JiST benefits from the automatic garbage collection, type-safety, reflection and many other properties of the Java language. This approach also lowers the learning curve for users and facilitates the reuse of code for building simulations. The use of a standard virtual machine provides an efficient, highly-optimized and portable execution platform and allows for important crosslayer optimization between the simulation kernel and running simulation. Furthermore, since the kernel and the simulation are both running within the same process space we reduce serialization and context switching overheads.

In summary, a key benefit of the JiST approach is that it allows for the efficient execution of simulation programs within the context of a modern and popular language. JiST combines simulation semantics, found in custom simulation languages and simulation libraries, with modern language capabilities. This design results in a system that is convenient to use, robust and efficient.

4.1.3. SWAN

SWANS is a Scalable Wireless Ad hoc Network Simulator built atop the JiST platform, a general-purpose discrete event simulation engine. SWANS [13] was created primarily because existing wireless network simulation tools are not sufficient for current research needs. SWANS also serves as a validation of the virtual machine-based approach to simulator construction.

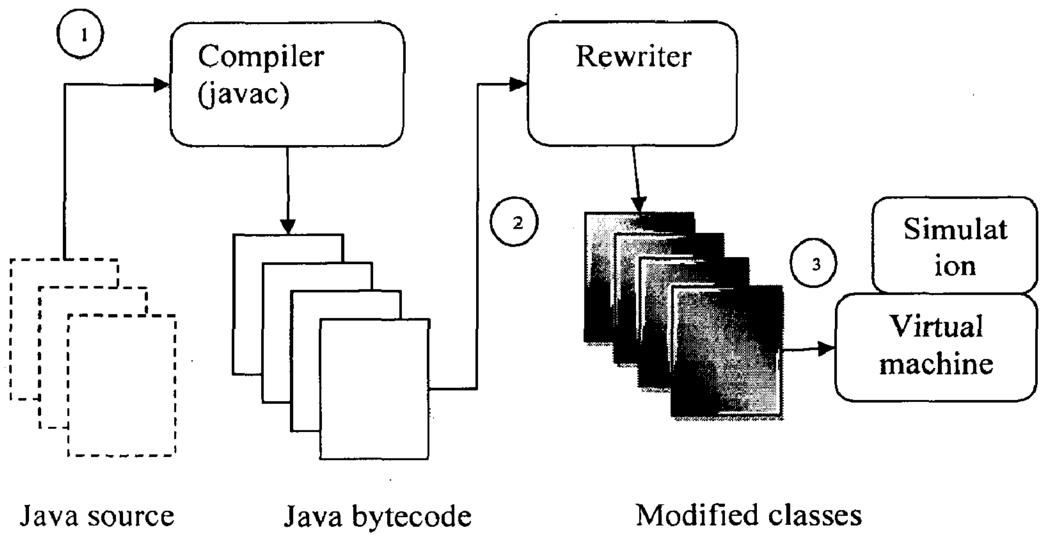


Figure 4.1. JiST architecture

SWANS [13] has a unique and important advantage over existing network simulators. It can run regular, unmodified Java network applications over the simulated network, thus allowing for the inclusion of existing Java-based software, such as web servers, peer-to-peer applications and application-level multicast protocols. These applications do not merely send packets to the simulator from other processes. They operate in simulation time within the same JiST process space, allowing far greater scalability.

4.1.3.1. Design highlights

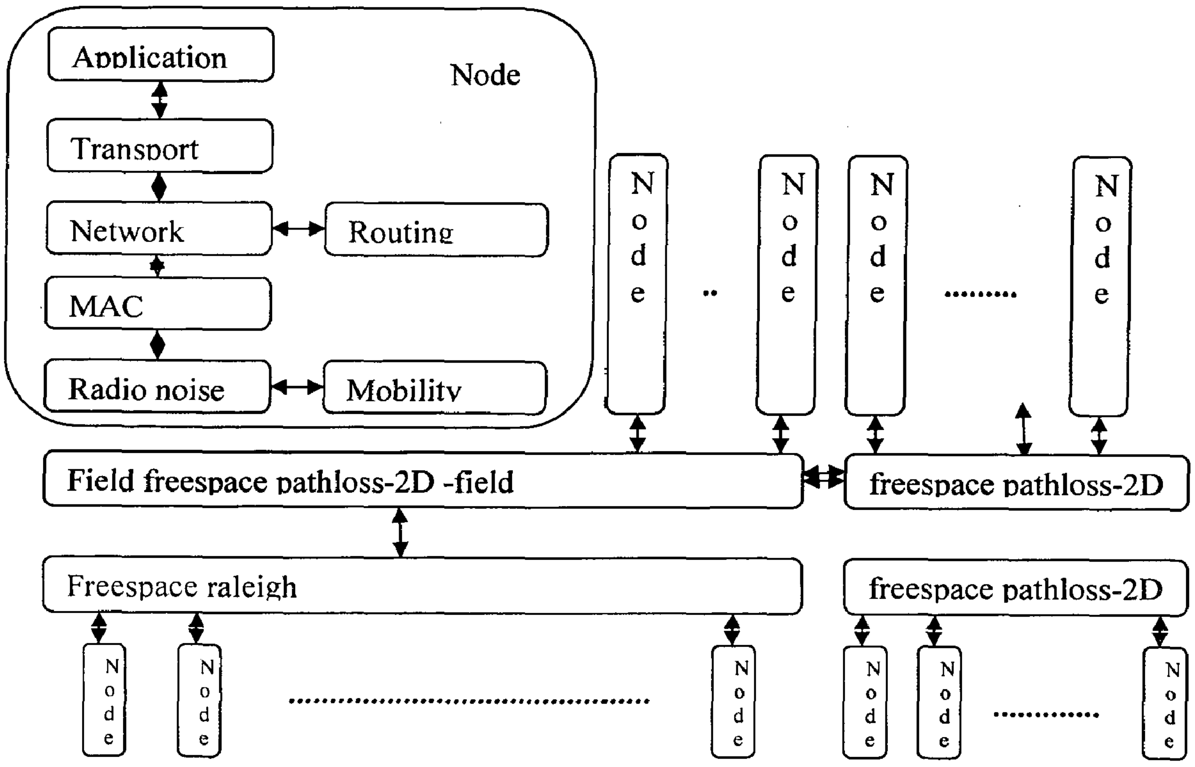
The SWANS software is organized as independent software components that can be composed to form complete wireless network or sensor network simulations, as shown in Figure 4.2. Its capabilities are similar to ns2 [17] and GloMoSim [18], two popular wireless network simulators. There are components that implement different types of applications; networking, routing and media access protocols; radio transmission, reception and noise models; signal propagation and fading models; and node mobility models. Instances of each component type are shown italicized in the Figure 4.2.

Notably, the development of SWANS has been relatively painless. Since JiST inter-entity message creation and delivery is implicit, as well as message garbage collection and typing, the code is compact and intuitive. Components in JiST consume less than half of the code (in uncommented line counts) of comparable components in GloMoSim, which are already smaller than their counterpart implementations in ns2. Every SWANS component is encapsulated as a JiST entity: it stores it own local state and interacts with other components via exposed event-based interfaces. SWANS contains components for constructing a node stack, as well components for a variety of mobility models and field configurations. This pattern simplifies simulation development by reducing the problem to creating relatively small, event-driven components. It also explicitly partitions the simulation state and the degree of inter-dependence between components, unlike the design of ns2 and GloMoSim. It also allows components to be readily interchanged with suitable alternate implementations of the common interfaces and for each simulated node to be independently configured. Finally, it also confines the simulation communication pattern. For example, Application or Routing components of different nodes cannot communicate directly. They can only pass messages along their own node stacks.

Consequently, the elements of the simulated node stack above the Radio layer become trivially parallelizable, and may be distributed with low synchronization cost. In contrast, different Radios do contend (in simulation time) over the shared Field entity and raise the synchronization cost of a concurrent simulation execution. To reduce this contention in a distributed simulation, the simulated field may be partitioned into non-overlapping, cooperating Field entities along a grid.

It is important to note that, in JiST, communication among entities is very efficient. The design incurs no serialization, copy, or context-switching cost among co-located entities, since the Java objects contained within events are passed along by reference via the simulation time kernel. Simulated network packets are actually a chain of nested objects that mimic the chain of packet headers added by the network stack. Moreover, since the packets are timeless by design, a single broadcasted packet can be safely shared among all the receiving nodes and the very same object sent by an Application entity on one node will be received at the Application entity of another node. Similarly, if we use TCP in our node stack, then the same object will be referenced in the sending node's TCP retransmit buffer.

This design conserves memory, which in turn allows for the simulation of larger network models.



4.2 SWAN architecture.

Results and Analysis

Chapter 5

5.1. Simulation Setup

To evaluate the performance of EAMR JiST-SWANS [12,13] is used. The simulation consists of a network of 35 nodes confined in a 800×800 m^2 area. Random connections were established using CBR traffic (at 4 packets/second with a packet size of 1024 bytes). The initial battery capacity of each node is 10 units. This initial energy is progressively reduced by data transmission/reception. When it reaches zero units, the corresponding node cannot take part any more in the communication, and is regarded as died. Each node has a radio propagation range of 250 meters and channel capacity was 2 Mb/s. Remaining parameter is as shown in following Table 5.1.

Table 5.1 Parameters Values used in Simulation

Routing Protocol	LEARAODV
Number of Nodes	35
Simulation Area	800×800 m^2
Transmission Range	250 meters
Initial Battery Capacity	10 units
Channel Capacity	2Mb/s
Connection Type	CBR
Packet Size	1024
Node Speed	2-8 m/sec
Mobility Model	Random Waypoint
PathLoss Model	Two-Ray

Spatial Model	Hierarchical Grid
Placement	Random
Fading Model	Zero Fading Model
Antenna Gain	15dB
Interference Model	RadioNoiseAdditive

5.2. Performance Metrics

We evaluate four performance metrics:

1. ***Route discovery frequency***: The total number of route discoveries initiated per second
2. ***Lifetime of node***: time at which node died
3. ***Average end-to-end delay of data packets***: This includes all possible delays caused by buffering during route discovery, queuing delay at the interface, retransmission delays at the MAC, propagation and transfer times;
4. ***Packet delivery fraction***: Ratio of the data packets delivered to the destination to those generated by the CBR sources; or a related metric *received throughput* in Kb/sec received at the destination;

In this performance evaluation we considered two factors, first is when nodes are stationary and other is when nodes are mobile.

5.3. Simulation results

Figures in this chapter shows comparison of AODV, AMODV, and LEAR-AODV with EAMR considering the four performance metrics as a function of mobility. Maximum speed of the nodes is varied from 0 m/s to 30 m/s to change mobility. Maximum speed of 0 m/s corresponds to a static network. Average rate of link failures in our mobility scenarios increases between 0–50 per second as the maximum speed increases between 0–30 m/s. The packet rate is 4 packets/sec. Performance of EAMR is more apparent at low as well as high speed (Figure5). As expected, the fraction of packets delivered goes down for all the protocols. However, EAMR loses fewer packets than others in mobile cases. There is a tremendous reduction in the average end-to end delay with EAMR as compared to AODV and LEAR-AODV as shown in Figure 5.2. Improvement in delay is almost always more than

100% compared to AODV and LEAR-AODV. This is because availability of alternate routes on route failures eliminates route discovery latency that contributes to the delay. Interestingly, for all protocols the delay increases with mobility only until the maximum speed of 10 m/s and beyond that delays stabilize. With additional instrumentation, we found that packet drops at intermediate nodes due to link failures beyond 10 m/s are dominated by packets with longer path lengths. In other words, the average hopcount of delivered data packets comes down at high speeds. Thus, delays become insensitive to increase in mobility after a point as the packets delivered at high speeds are mostly those that travel along shorter paths. As expected, EAMR performs better in all the metrics.

Figure 5.2. Shows that when nodes are stationary, EAMR gives better results as compared to other protocols, it means that when speed of node is 0 meters/second then frequency of route discovery procedure in AODV, LEAR-AODV and AMODV is 0.8,0.7 and 0.5 respectively, where 0.2 in case of EAMR. In high mobility environment, EAMR's performance regarding to route discovery frequency is near about AOMDV which is quite significant.

Figure 5.2. and Figure 5.3. shows the time instances at which certain number of nodes has died because of their batteries depletion when nodes are stationary and are mobile respectively. We note that when nodes are stationary for Energy Conservative Multipath Routing, the first node dies approximately 2986 seconds later than in AODV, 1111 seconds later than in LEAR-AODV, and 2990 seconds later than in AOMDV. The effect of mobility is shown in Figure 5.1. We also noted that when node velocity equal to 5 meters/second, for Energy e Multipath Routing the first node dies approximately 2703 seconds later than in AODV, 1520 seconds later than in LEAR-AODV, and 2732 seconds later than in AOMDV. However, as the velocity of the node movement increases, rate of energy consumption in the network goes up. This is normal since higher velocity of movement implies more route discoveries being performed and as a consequence higher energy consumption in the network.

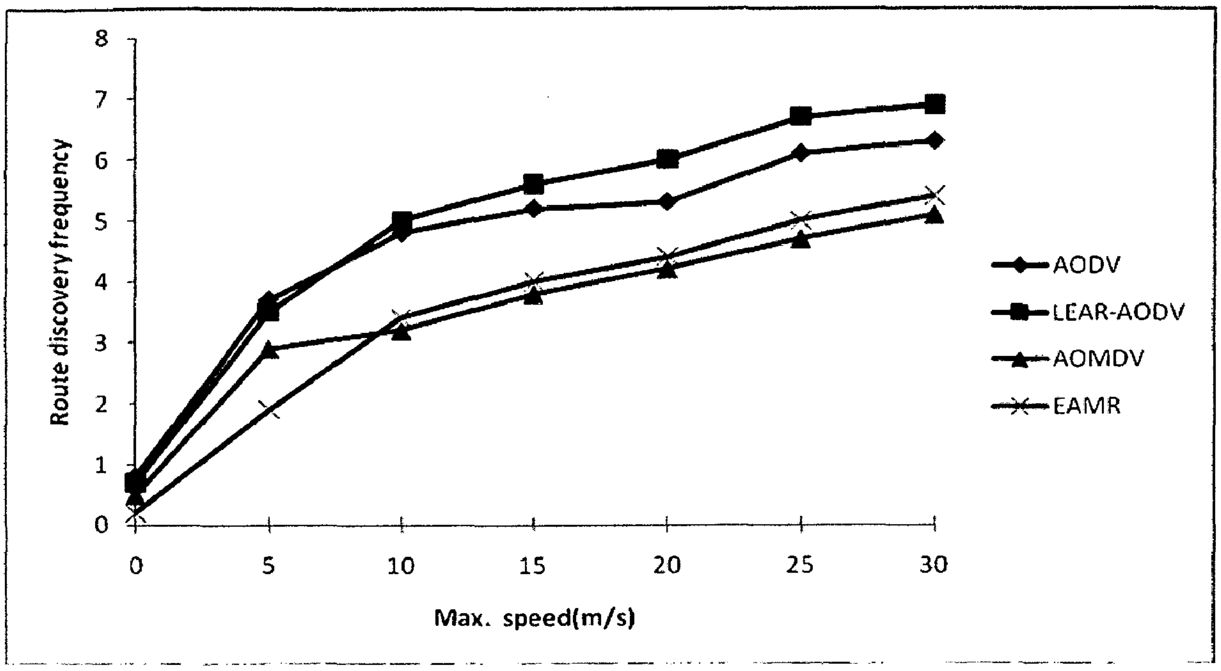


Figure 5.1 Frequency of route discovery procedure in meter per second

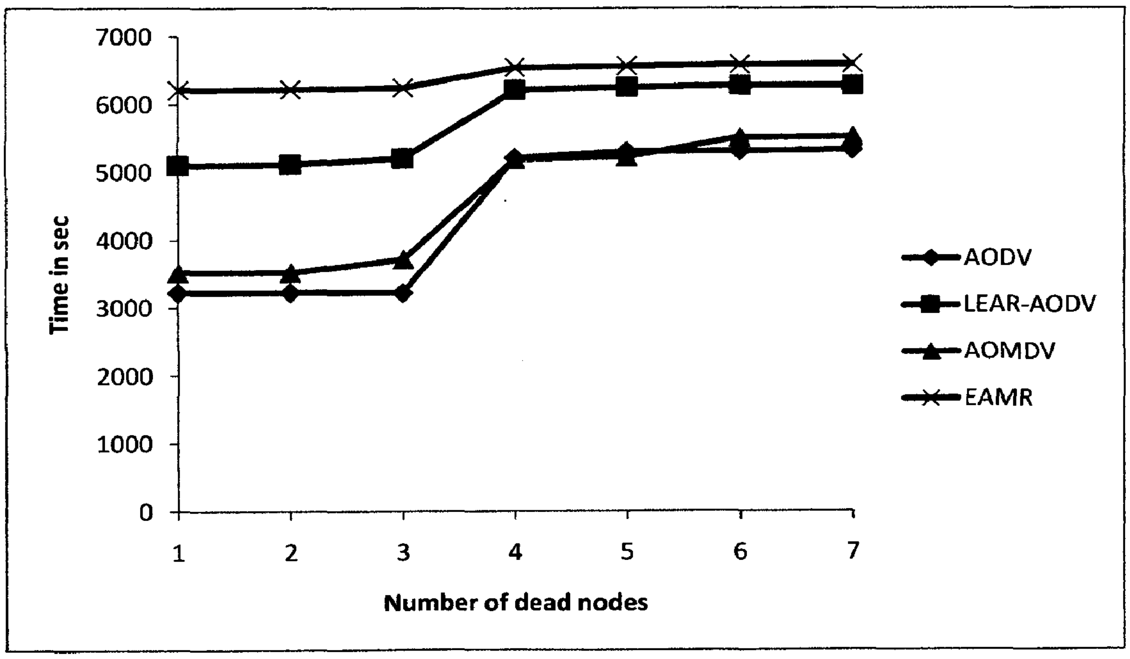


Figure 5.2. The number of dead nodes versus time when nodes are stationary

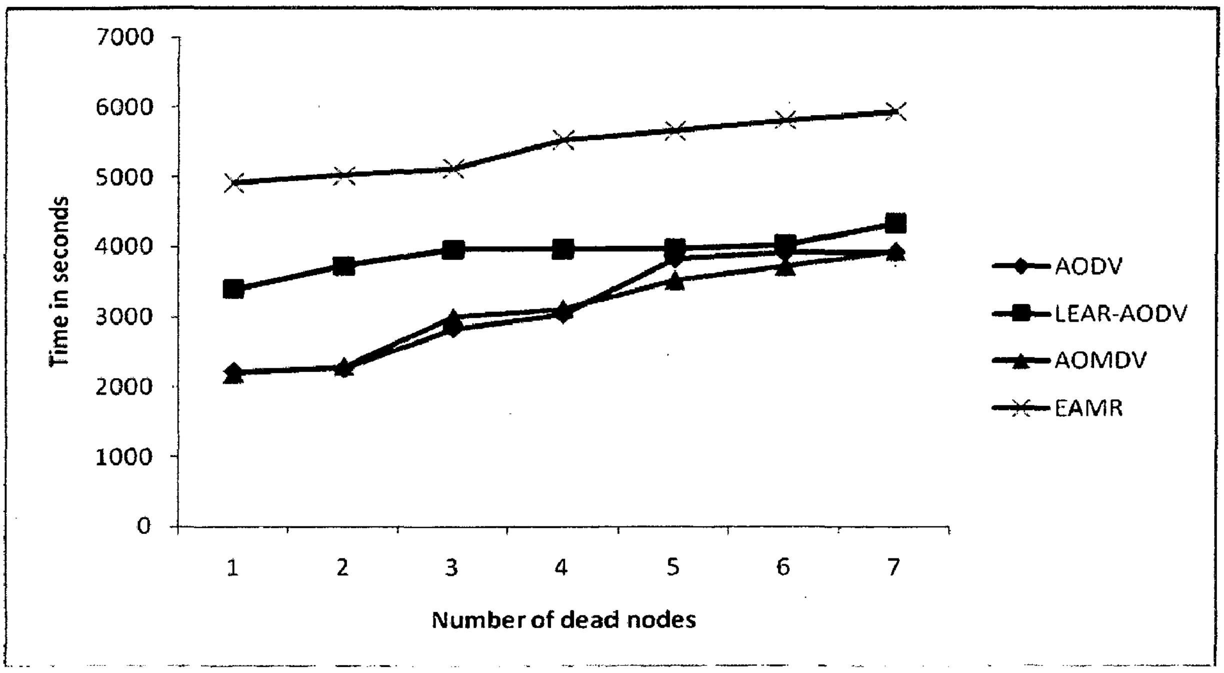


Figure 5.3. The number of dead nodes versus time when nodes are mobile (5 m/s)

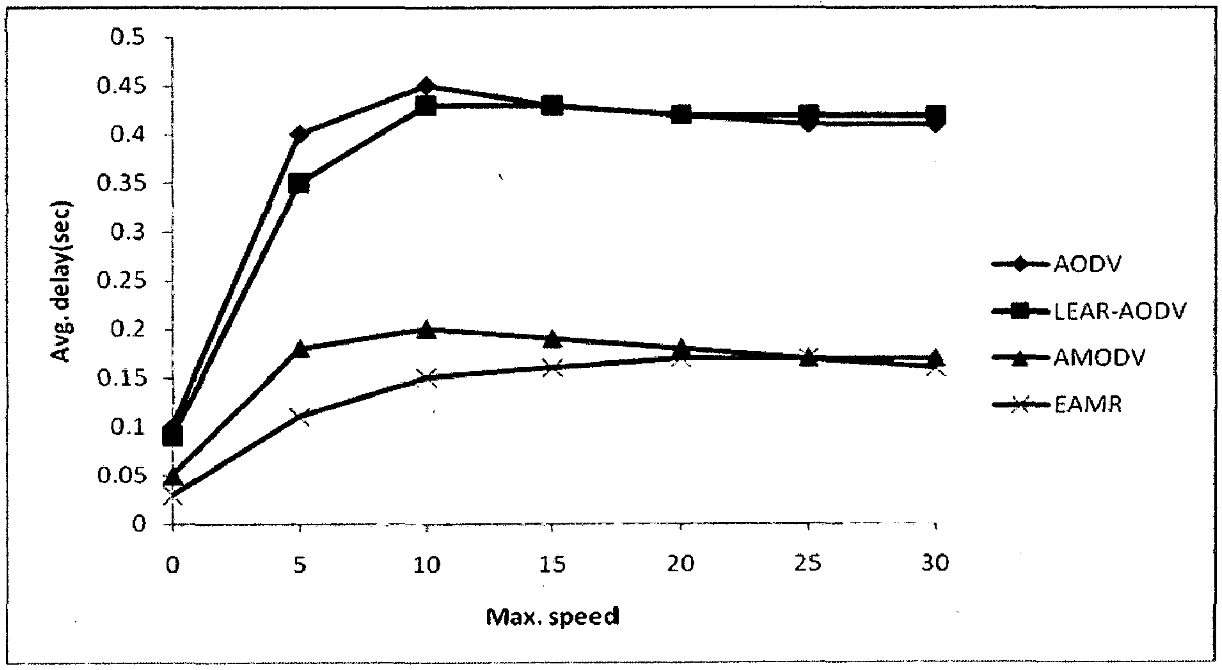


Figure 5.4. Average delay with varying speed.

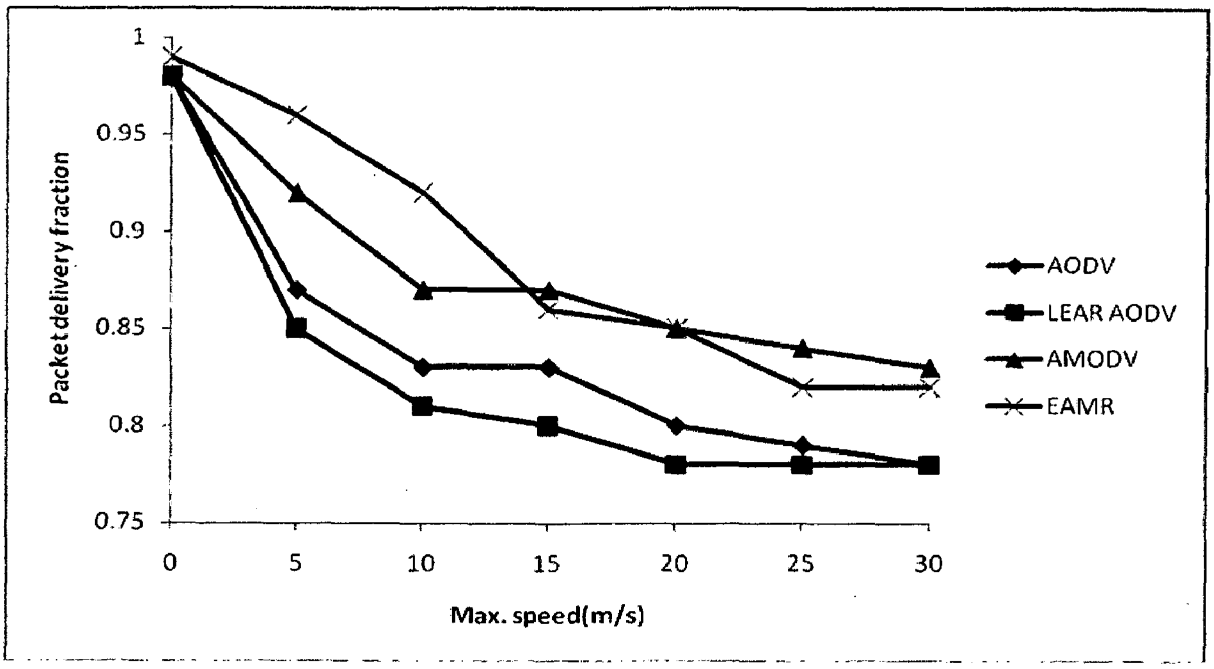


Figure 5.5. Data packet delivery fraction.

Conclusion

Chapter 6

Wireless mobile adhoc network present difficult challenges to routing protocol designers. Mobility and limited power cause frequent topology changes which effect the performance of network. So battery power and effect of mobility are major issues in mobile ad hoc network (MANET) and should be considered while routing.

In this work we proposed and analyzed an Energy aware multipath routing protocol (EAMR) for MANETs where goal is to conserve battery power and reduce route discovery frequency in routing process. There are two main contributions of this work.

1. We maintain multiple paths for better connectivity and to reduce routing overhead.
2. We balanced individual node's battery power utilization to prolong the entire network's lifetime.

We have studied the performance of EAMR relative to other protocols under a wide range of mobility and traffic scenarios. We observe that EAMR performs well in both stationary and dynamic conditions.

Though EAMR is resulting significantly well, yet there is scope of better performance. In future work, there is a possibility of enhancement of EAMR by implementing energy awareness in distributed manner with consideration of other issues related to on demand multipath routing, for example, load balancing with multiple paths.

References

[1] IETF MANET WG (Mobile Ad hoc Network), www.ietf.org/html.charters/manet-charter.html.

[2] E. M. Royer and C-K Toh, "A Review of Current Routing Protocols for Ad hoc Wireless Networks", IEEE Personal Communication, pp. 46-55, April 1999,.

[3] S. Bahk and W. El-Zarki, "Dynamic Multi-path Routing and how it Compares with other Dynamic Routing Algorithms for High Speed Wide-area Networks", in Proc. of the ACM SIGCOM, December. 1992

[4] E. Perkins, E. M. Belding-Royer, and S. R.Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", IETF Internet Draft, draft-ietf-manet-aodv-13.txt

[5] J. Broch, D. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols", In Proceedings of the IEEE/ACM MOBICOM, pages 85–97, 1998.

[6] S. R. Das, R. Castaneda, and J. Yan, "Simulation-based Performance Evaluation of Routing Protocols for Mobile Ad hoc Networks". ACM/Baltzer Mobile Networks and Applications (MONET), 5(3):179–189, June 2000.

[7] P. Johansson, T. Larsson, N. Hedman, and B. Mielczarek, "Routing Protocols for Mobile Ad-hoc Networks – A Comparative Performance Analysis". In Proceedings of the IEEE/ACM MOBICOM, pages 195–206, April 1999.

[8] M.K. Marina, S.R. Das, "On-demand Multipath Distance Vector Routing in Ad Hoc Networks," in Proceedings of the International Conference for Network Procotols, August 2001.

[9] S. Singh, M.Woo, and C. S. Raghavendra. "Power-aware Routing in Mobile Ad Hoc Networks", In Proceedings of ACM MobiCom, pp. 181–190, December 1998.

[10] L. Ouakil, S. Senouci, and G. Pujolle, "Performance Comparison of Ad Hoc Routing Protocols Based on Energy Consumption", Ambience Workshop 2002, Torino, Italy,September 2002.

[11] S.-M. Senouci and M. Naimi, "New routing for balanced energy consumption in mobile ad hoc networks," in PE-WASUN '05: Proceedings of the 2nd ACM international

workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks. New York, NY USA: ACM Press, pp. 238–241, February 2005.

[12] Rimon Barr, JiST–Java in Simulation Time User Guide, http://jist.ece.cornell.edu/docs/040319-jist-user.pdf.

[13] Rimon Barr, SWANS– Scalable Wireless Ad hoc Network Simulator User Guide, http://jist.ece.cornell.edu/docs/040319-swans-user.pdf

[14] Se-Won Jung ,Chae-Woo Lee, "Energy-Aware Routing Algorithm Using Backup Route for Ad-Hoc Networks" international conference on Computational Science and its Applications ,Glasgov, ROYAUME UN, February 2006.

[15] M.Mleki,K. Dantu, and M. Pedram, "Power Aware Source Routing in Mobile Ad Hoc Network" Proceedings of ISLPED'02,Monterey.CA.pp.72-75, August 2002.

[16] M. Maleki,K. Dantu,and M. Pedram, "Lifetime Prediction Routing in Mobile Ad Hoc Networks",IEEE Wireless Communication and Network Conference , Mars, 2003.

[17] S. McCanne and S. Floyd. ns (Network Simulator) at http://www-nrg.ee.lbl.gov/ns, 1995.

[18] X. Zeng, R. L. Bagrodia, and M. Gerla. GloMoSim: a library for parallel simulation of large-scale wireless networks. In *PADS*, May 1998.

[19] Seung Jun Baek , Gustavo de Veciana, Spatial energy balancing through proactive multipath routing in wireless multihop networks, IEEE/ACM Transactions on Networking (TON), v.15 n.1, p.93-104, February 2007.

[20] Ian F. Akyildz, Won-Yeol Lee, kaushik R. Chowdhary "Cognitive radio ad hoc networks", Elsevier Science, Ad Hoc Networks, vol.7, issue 5, 2009.

[21] Demistrios, Y. Charlie Hu, "Exploring the design space of reliable multicast protocols for wireless mesh networks" Elsevier Science, Ad Hoc Networks, vol.7, issue 5,2009.

[22] Shi Zheng, Weiqiang Wu, Qinyu Zhang, "Energy-aware and level-distribution routing protocol in ad hoc networks", ICAIT '08: Proceedings of the 2008 International Conference on Advanced Infocomm Technology, July 2008.

[23] Dengfeng Yang, Xueping Li, Rapinder Sawhney, Xiaorui Wang, "Geographic and energy-aware routing in Wireless Sensor Networks", International Journal of Ad Hoc and Ubiquitous Computing, Volume 4 Issue 2, March2009.

1. Mangesh R. Mane, A.K. Sarje, "Local Energy Conservative Multipath Routing Protocol for Mobile Adhoc Networks ", Accepted in International Conference on Industrial and Information Systems, NIT Suratkal , Aug 2010.

Appendix A

TTL Calculation and Configuration of Parameters

TTL Calculation

To prevent unnecessary network-wide dissemination of RREQs, the originating node should use an expanding ring search technique. In an expanding ring search, the originating node initially uses a TTL = TTL_START in the RREQ packet IP header and sets the timeout for receiving a RREP to RING_TRAVERSAL_TIME milliseconds The TTL_VALUE used in calculating RING_TRAVERSAL_TIME is set equal to the value of the TTL field in the IP header. If the RREQ times out without a corresponding RREP, the originator broadcasts the RREQ again with the TTL incremented by TTL_INCREMENT. This continues until the TTL set in the RREQ reaches TTL_THRESHOLD, beyond which a TTL = NET_DIAMETER is used for each attempt. Each time, the timeout for receiving a RREP is RING_TRAVERSAL_TIME. When it is desired to have all retries traverse the entire ad hoc network, this can be achieved by configuring TTL_START and TTL_INCREMENT both to be the same value as NET_DIAMETER.

The Hop Count stored in an invalid routing table entry indicates the last known hop count to that destination in the routing table. When a new route to the same destination is required at a later time (e.g., upon route loss), the TTL in the RREQ IP header is initially set to the Hop Count plus TTL_INCREMENT. Thereafter, following each timeout the TTL is incremented by TTL_INCREMENT until TTL = TTL_THRESHOLD is reached. Beyond this TTL = NET_DIAMETER is used.

Once TTL = NET_DIAMETER, the timeout for waiting for the RREP is set to NET_TRAVERSAL_TIME,

Configuration of Parameters

Table A.1. gives default values for some important parameters associated with EAMR protocol operations. A particular mobile node may wish to change certain of the parameters, in particular the NET_DIAMETER, MY_ROUTE_TIMEOUT, ALLOWED_HELLO_LOSS, RREQ_RETRIES, and possibly the HELLO_INTERVAL. In the latter case, the node should advertise the HELLO_INTERVAL in its Hello messages, by appending a Hello Interval Extension to the RREP message. Choice of these parameters may affect the performance of the protocol. Changing NODE_TRAVERSAL_TIME also changes the node's estimate of the NET_TRAVERSAL_TIME, and so can only be done with suitable knowledge about the behavior of other nodes in the ad hoc network. The configured value for MY_ROUTE_TIMEOUT MUST be at least 2 * PATH_DISCOVERY_TIME.

Table A.1. Default values for important parameters

Parameter Name	Value
ACTIVE_ROUTE_TIMEOUT	3,000 Milliseconds
ALLOWED_HELLO_LOSS	2
BLACKLIST_TIMEOUT	RREQ_RETRIES * NET_TRAVERSAL_TIME
LOCAL_ADD_TTL	2
MAX_REPAIR_TTL	0.3 * NET_DIAMETER
NET_TRAVERSAL_TIME	2 * NODE_TRAVERSAL_TIME * NET_DIAMETER
NEXT_HOP_WAIT	NODE_TRAVERSAL_TIME + 10
NODE_TRAVERSAL_TIME	40 milliseconds
PATH_DISCOVERY_TIME	2 * NET_TRAVERSAL_TIME
RING_TRAVERSAL_TIME	2 * NODE_TRAVERSAL_TIME * (TTL_VALUE + TIMEOUT_BUFFER)
TTL_START	1
TTL_INCREMENT	2
TTL_THRESHOLD	7