A comparative study of routing protocols in MANETs

Muhammad Ali
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd

Recommended Citation
https://lib.dr.iastate.edu/rtd/19890

This Thesis is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
A comparative study of routing protocols in MANETs

by

Muhammad Ali

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Engineering

Program of Study Committee:
Mani Mina, Co-major Professor
Doug Jacobson, Co-major Professor
Thomas J. Rudolphi

Iowa State University
Ames, Iowa

2003

Copyright © Muhammad Ali, 2003. All rights reserved.
Graduate College
Iowa State University

This is to certify that the Master’s thesis of
Muhammad Ali
has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy
I would like to dedicate this thesis to my wife Zahra and to my sons Husnain, Saqlain and daughter Aainy without whose support I would not have been able to complete this work. I would also like to thank my friends and family for their loving guidance, and support during the writing of this work.
# TABLE OF CONTENTS

**LIST OF FIGURES** ........................................... vii

**ABSTRACT** ................................................. ix

**CHAPTER 1. INTRODUCTION** ............................... 1

  1.1 Mobile Ad Hoc Networks ............................... 1
  1.2 Overview of Flat Routing Protocols .................. 2
    1.2.1 Proactive Table-Driven Protocols ................ 3
    1.2.2 Reactive On-demand Protocols .................... 4
  1.3 Overview of Thesis .................................... 5

**CHAPTER 2. ADHOC NETWORKS** ............................ 6

  2.1 Wireless Medium ...................................... 6
    2.1.1 Infrared ........................................ 7
    2.1.2 Microwave ....................................... 7
    2.1.3 Radio ........................................... 8
    2.1.4 Medium Access ................................... 9
  2.2 Routing Protocols ................................... 10
    2.2.1 Basics Of Routing Protocols ..................... 11
  2.3 Associativity Based Routing(ABR) ..................... 12
    2.3.1 Ad hoc On Demand Distance Vector Protocol (AODV) 13
    2.3.2 Bordercast Resolution Protocol (BRP) ............. 13
    2.3.3 Core Extraction Distributed Ad hoc Routing (CEDAR) 14
2.3.4 Cluster Based Routing Protocol (CBRP) ........................................ 15
2.3.5 Dynamic Source Routing Protocol (DSR) ................................. 16
2.3.6 DSR-Flow protocol .............................................................. 16
2.3.7 Fisheye State Routing Protocol (FSR) ..................................... 17
2.3.8 Host Specific Routing (HSR) .................................................... 17
2.3.9 Intrazone Routing Protocol (IARP) ......................................... 18
2.3.10 Interzone Routing Protocol (IERP) ........................................ 18
2.3.11 An Internet MANET Encapsulation Protocol (IMEP) .............. 19
2.3.12 Landmark Routing Protocol for Large Scale Networks (LANMAR) 19
2.3.13 Lightweight Underlay Network Ad hoc Routing (LUNAR) ......... 20
2.3.14 Mobile Mesh Border Discovery Protocol (MMBDP) ................. 20
2.3.15 Multicast Zone Routing protocol (MZR) ................................. 21
2.3.16 Optimized Link State Routing Protocol (OLSR) ..................... 21
2.3.17 Relative Distance Micro-Discovery Ad Hoc Routing (Rdmar) Protocol .............................................................. 22
2.3.18 Source Tree Adaptive Routing Protocol (STAR) ..................... 23
2.3.19 Topology Broadcast Reverse-Path Forwarding Protocol (TBRPF) 23
2.3.20 Temporally-Ordered Routing Algorithm Protocol (TORA) ....... 24
2.3.21 Zone Routing Protocol (ZRP) ................................................ 24

CHAPTER 3. NETWORK SIMULATOR ................................................. 25

3.1 Introduction ............................................................................. 25
3.1.1 Network Simulator NS2 ........................................................ 25
3.1.2 Optimal NS2 Learning Strategy ............................................. 26
3.1.3 Simulation Difficulties .......................................................... 28
3.1.4 Creating Random Traffic-Pattern For Wireless Scenarios. ....... 30
3.1.5 Creating Node-Movements For Wireless Scenarios. ............... 32
CHAPTER 4. IMPLEMENTATION OF ADHOC ROUTING PROTOCOLS ........................................................................... 35

4.1 Destination Sequenced Distance Vector (DSDV) ......................................................... 35
4.2 Temporally Ordered Routing Algorithm (TORA) ......................................................... 38
4.3 Ad Hoc On-Demand Distance Vector (AODV) ............................................................. 42
4.4 Simulation Model ........................................................................................................... 44
   4.4.1 Network Simulator .................................................................................................. 45
   4.4.2 Physical Layer Model ............................................................................................ 45
   4.4.3 Simulation Overview .............................................................................................. 48
4.5 Performance Evaluation Criteria .................................................................................... 49
4.6 Parsing and Analyzing Data .......................................................................................... 50
4.7 Performance Evaluation ................................................................................................. 52

CHAPTER 5. CONCLUSION AND FUTURE WORKS ............................................................. 61

5.1 Conclusions ................................................................................................................... 61
5.2 Future Work ................................................................................................................... 63

APPENDIX A. MOBILITY MODELS USED IN SIMULATIONS ............................................ 64

APPENDIX B. LEARNING NS-2 ............................................................................................. 68

APPENDIX C. OTHER NETWORK SIMULATORS ............................................................... 70

BIBLIOGRAPHY ...................................................................................................................... 73

ACKNOWLEDGEMENTS ........................................................................................................ 85
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Classification of Ad Hoc Routing Protocols</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>A Simple Ad Hoc Network</td>
<td>6</td>
</tr>
<tr>
<td>3.1</td>
<td>Directory Tree of NS2</td>
<td>28</td>
</tr>
<tr>
<td>4.1</td>
<td>DSDV propagation of route information for destination MH8.</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>Each tuple represents the destination address, sequence number, and hop count</td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td>Directed acyclic graph having destination D, water flow analogy</td>
<td>39</td>
</tr>
<tr>
<td>4.3</td>
<td>Route creation in TORA</td>
<td>40</td>
</tr>
<tr>
<td>4.4</td>
<td>TORA, when route is broken</td>
<td>41</td>
</tr>
<tr>
<td>4.5</td>
<td>Route discovery in AODV</td>
<td>43</td>
</tr>
<tr>
<td>4.6</td>
<td>Generalized Simulation Method</td>
<td>44</td>
</tr>
<tr>
<td>4.7</td>
<td>Hidden terminal problem that occurs when node A unwittingly interferes with</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>the transmission of node C</td>
<td></td>
</tr>
<tr>
<td>4.8</td>
<td>Schematic of a mobile node in NS2, featuring DSDV agent</td>
<td>48</td>
</tr>
<tr>
<td>4.9</td>
<td>Simulation process</td>
<td>49</td>
</tr>
<tr>
<td>4.10</td>
<td>Packet Delivery Fraction(a) - 10 Sources</td>
<td>52</td>
</tr>
<tr>
<td>4.11</td>
<td>Packet Delivery Fraction(b) - 25 Sources</td>
<td>53</td>
</tr>
<tr>
<td>4.12</td>
<td>Packet Delivery Fraction(c) - 5 Sources</td>
<td>56</td>
</tr>
<tr>
<td>4.13</td>
<td>Packet Delivery Fraction(d) - 15 Sources</td>
<td>56</td>
</tr>
</tbody>
</table>
Figure 4.14  Average End to End Delay(a) - 10 Sources  .................  57
Figure 4.15  Average End to End Delay(b) - 25 Sources  .................  57
Figure 4.16  Average End to End Delay(c) - 5 Sources  .................  58
Figure 4.17  Average End to End Delay(db) - 15 Sources  ...............  58
Figure 4.18  Normalized Routing Load(a) - 10 Sources  ..................  59
Figure 4.19  Normalized Routing Load(b) - 25 Sources  ..................  59
Figure 4.20  Normalized Routing Load(c) - 5 Sources  ....................  60
Figure 4.21  Normalized Routing Load(d) - 15 Sources  ....................  60
Mobile Ad Hoc networks are emerging area of mobile computing. A “mobile ad hoc network” (MANET) is composed of mobile routers and associated hosts connected by wireless links. The routers are free to move randomly and organize themselves arbitrarily, thus, the network's wireless topology may change rapidly and unpredictably. In fact, it is considered that each node would have some capacity to relay the information thus constrained by computational power, battery life and increasingly complex routing with added functionality of a router. Nodes may keep joining and leaving an ad hoc network. Such a network may operate in a stand alone fashion, or may be connected to the larger Internet.

Lack of infrastructure in ad hoc networks sets new challenges for routing algorithms where the network is formed by a collection of wireless mobile nodes dynamically forming a temporary network without the use of any existing network infrastructure or centralized administration. A number of routing protocols like Dynamic Source Routing (DSR), Ad Hoc On-Demand Distance Vector Routing (AODV), Destination-Sequenced Distance-Vector (DSDV), Zone Routing Protocol (ZRP) and Temporally Ordered Routing Algorithm (TORA) have been implemented. In this thesis an attempt has been made to compare the performance of prominent on-demand reactive routing protocols for mobile ad hoc networks (AODV and TORA), along with the traditional proactive DSDV protocol. Although AODV and TORA share similar on-demand behavior, the differences in the protocol mechanics can lead to significant performance differentials. The performance differentials are analyzed using varying network loads, mobilities, and
network sizes. These simulations are carried out using network simulator (ns-2.1b9a) to run mobile ad hoc network simulations.
CHAPTER 1. INTRODUCTION

1.1 Mobile Ad Hoc Networks

Ad Hoc networks are key to the evolution of wireless networks. Ad Hoc networks are typically composed of nodes that communicate over wireless links without any central command. Mobile Ad Hoc Networks (MANETs) are an emerging technology that allows establishing instant communication infrastructures for civilian and military applications. Target applications range from collaborative, distributed mobile computing (sensors, conferences, conventions) to disaster recovery (such as fire, flood, earthquake), law enforcement (crowd control, search and rescue) and tactical communications (digital battlefields). An ad hoc network is self-organizing and communicates mostly through multi hop wireless links. Mobility of network members, limited resources (e.g., bandwidth and energy supply) and potentially large number of mobile nodes make routing in ad hoc networks extremely challenging. Routing protocols generally use either distance-vector or link-state routing algorithms (2). Both types find shortest paths to destinations. In distance-vector routing (DV), a vector containing the cost (e.g., hop distance) and path (next hop) to all the destinations is kept and exchanged at each node. DV protocols are generally known to suffer from slow route convergence and a tendency to create loops in mobile environments. The link-state routing (LS) algorithm overcomes the problem by maintaining global network topology information at each router through periodical flooding of link information about its neighbors. Mobility entails frequent flooding. Unfortunately, this LS advertisement scheme generates larger routing control
over-head than DV. In a network with population N, LS updating generates routing over-
head on the order of $O(N^2)$. In large networks, the transmission of routing information
will ultimately consume most of the bandwidth and consequently block applications,
rendering it unfeasible for bandwidth limited wireless ad hoc networks. We can classify
various protocols according to the network structure underlying routing protocols as
shown in Figure 1.1. In the subsequent chapters, we will review some of the key routing
protocols in Ad Hoc networks.

Flat routing approaches adopt a flat addressing scheme. Each node participating
in routing plays an equal role. In contrast, hierarchical routing usually assigns differ-
ent roles to network nodes. Some protocols require a hierarchical addressing system.
Routing with assistance from geographic location information requires each node to be
equipped with the Global Positioning System (GPS)(27). This requirement is quite re-
alistic today since such devices are reasonably priced and can provide required precision.
The thesis concludes with a comparison of salient protocols (DSDV, AODV and TORA)
in the category of proactive and reactive protocols and future research directions.

1.2 Overview of Flat Routing Protocols

The protocols here fall into two main categories: proactive and reactive. Many of
the proactive protocols use conventional LS routing. On-demand routing, on the other
hand uses a different philosophy for ad hoc routing. It differs from conventional routing
protocols, in which no routing activities and no permanent routing information are
maintained at the nodes if there is no communication, thus providing a scalable routing
solution to large populations.
1.2.1 Proactive Table-Driven Protocols

In proactive protocols, each node stores routing information for every other node in the network. These protocols are akin to traditional routing protocols in that they periodically distribute route information to keep all nodes up to date. Each router maintains a table (or tables) to determine the next hop for a packet, given its destination address. The main difference between proactive protocols for ad hoc networks and traditional protocols is that the former assumes all nodes will participate as routers on the network. The goal of these table-driven protocols is to converge on the optimal paths for all destinations on the network. Based on how they determine the "optimal" next hop for a given destination, proactive protocols for MANET environments fall into one of two categories: link-state and distance-vector (5).

Protocols following the link-state approach maintain network topology information, associating a cost for each link. Each node periodically broadcasts its outgoing link costs...
to all other nodes. When nodes receive new link cost information, they update their map of the network topology and apply a shortest path algorithm to choose its next hop for each destination.

Distance-vector approaches are based on the classical Bellman-Ford algorithm (1). Each node maintains a list of distances from each of its neighbors to the intended destination. A node chooses the neighbor with the shortest distance to the destination as the next hop. These protocols must incorporate a mechanism to prevent routing loops, which are a well known problem with the Bellman-Ford algorithm (1).

1.2.2 Reactive On-demand Protocols

On-demand routing is a popular routing category for wireless ad hoc routing. The design follows the idea that each node tries to reduce routing overhead by only sending routing packets when a communication is awaiting. Examples include Ad Hoc On Demand Distance Vector Routing (AODV) (9), Associativity-Based Routing (ABR) (13), Dynamic Source Routing (DSR) (10) and (14), Lightweight Mobile Routing (LMR) (15), and Temporally Ordered Routing Algorithms (TORA) (16). Among the many proposed protocols, AODV and DSR have been extensively evaluated in the MANET literature and are being considered by the Internet Engineering Task Force (IETF) MANET Working Group as the leading candidates for standardization. They are described briefly here to demonstrate the on-demand routing mechanism. Interested readers are referred to (4),(12) and (10) for performance evaluation. On-demand algorithms typically have a route discovery phase. Query packets are flooded into the network by the sources in search of a path. The phase completes when a route is found or all the possible outgoing paths from the source are searched. There are different approaches for discovering routes in on-demand algorithms.
1.3 Overview of Thesis

A number of networking contexts, applicable to MANET routing protocols, exist today. Scenarios include conferences/meetings/lectures (5), emergency services (5), law enforcement (5), military communications, and embedded/wearable computers. Each of the scenarios present different network contexts based on network size, density, rate of topology transience, link capacity, link connectivity (synchronous or asynchronous), and traffic patterns. The applications associated with these scenarios may also imply different requirements in terms of security, quality of service (QoS) and power consumption. It is unclear whether a single MANET routing protocol will emerge to address all of these scenarios, or a suite of specialized protocols will address the different network contexts. In this thesis I have made an attempt to compare and evaluate the performance of DSDV, AODV and TORA routing protocols over a MANET.

Chapter 2 provides some introduction to Ad Hoc networks and wireless media along with a brief introduction to a number of protocols being currently studied in the literature. Most of the study focus on various aspects of the protocols, which include power saving features, QoS, effective routing and security issues. Chapter 3 gives a brief introduction to NS-2 with reference to Ad Hoc Routing Protocols and associated simulation difficulties. Chapter 4 provides detailed description of the protocols being simulated in this study and discusses simulation models and performance metrics used to evaluate a particular protocol with other related implementation issues in NS-2 environment. The results are also presented and evaluated in this chapter. Chapter 5 concludes this thesis discussing suggestions for future work in this area.
CHAPTER 2. ADHOC NETWORKS

In this section, general concepts of basic wireless communications and routing protocols for ad hoc networks are studied. A simple Ad hoc network is shown in Figure 2.1. Most of the proposed Ad Hoc Routing Protocol drafts are removed from IETF's site when they are not continually supported, consequently, finding detailed protocol information become difficult. In this chapter a limited review to briefly introduce some of the protocols based on the drafts and available literature is presented. A more detailed discussion on the protocol and simulation used in this study is provided in chapter 4.

2.1 Wireless Medium

There have been industrial, scientific, medical (ISM) wireless local area networks operating at 900 MHz, 2, 4 and 5 GHz since 1990s. However in 1998 IEEE approved
an international inter-operability standard IEEE 802.11 (17). The standard specifies Medium Access Control (MAC) protocol procedures and three types of Physical Layers (PHY). Two of them use radio, one uses infrared. All physical layers support 1 or 2 Mb/s data rates. Later in July 1998, as an extension, an 11 Mb/s Physical layer running on standard MAC was developed.

Smart or adaptive antennas for mobile communications have been of great interest, due to their provision of opportunity for increased bandwidth and range. Major drawback of such systems are transceiver complexity and more complex radio resource management.

2.1.1 Infrared

Infrared physical layer is simple and inexpensive. It uses the same signal frequencies as on fiber optic links. Infrared systems detect only the amplitude of the signal so the interference is reduced greatly. Bandwidth is not limited or restricted, so transmission speeds can be larger than other systems. Since it operates in the light spectrum, it does not require permission from FCC. This is probably the most attractive realm of infrared. When aimed, range increases to a few kilometers. For indoor usage, omnidirectional transmitters can be used, and a multi-hop network can be built. The only requirement is the true line of sight. It is cheap, however its spectrum is shared with the sun and fluorescent lights, and they cause heavy interference.

2.1.2 Microwave

Microwave operates at less than 500 mW (18). It uses narrow-band transmission with single frequency modulation at 5.8 GHz band. Microwave achieves higher throughput, because it does not involve the overhead of spread spectrum systems, such as DSSS system discussed in the subsequent paragraph. RadioLAN (19) is an example to microwave technology.
2.1.3 Radio

2.1.3.1 Direct Sequence Spread Spectrum (DSSS)

Direct sequence spread spectrum spreads the transmission signal over an allowed band (for example 25 MHz). A random binary string which is the spreading code, is used to modulate the transmitted signal. The data bits are mapped to a pattern of chips and mapped back into a bit sequence at the destination. The number of chips that represent a bit is equal to the spreading ratio. Higher the spreading ratio, the more the signal is resistant to the interference. IEEE 802.11 standard requires spreading ratio of 11 (20). The transmitter and receiver must be synchronized with the same spreading code. The use of orthogonal spreading codes allow utilization of more than one LAN on the same band. There is a study on distributed assignment of codes for multi-hop packet-radio networks which has an implementation of code assignment algorithm as a part of the Medium Access Controller (MAC)(21). However, since DSSS systems use wide sub-channels, the number of adjacent LANs is limited by the size of sub-channels. Moreover, recovery from interference is fast in DSSS, because of the ability to spread the signal over a wider band. WaveLAN (22) is an example DSSS product by Lucent Technologies (23).

2.1.3.2 Frequency Hopping Spread Spectrum (FHSS)

When we split the band into small sub-channels like 1 MHz, the signal may hop from one sub-channel to another transmitting short bursts of data on each for a set period of time, which is called dwell time. The hopping sequence must be synchronized at the sender and receiver nodes, or the data will be lost. Frequency hopping is less vulnerable to the interference, because frequency is always shifting. It is also difficult to intercept a frequency hopping communication. This is a favorable characteristic from security point of view. Attacks are only possible by jamming the whole band. Since sub-channels are
smaller than DSSS, more LANs can run simultaneously in the same band.

2.1.3.3 Multipath Interference

When a signal bounces off the walls or other barriers, interferences can be generated as they reach destination at different times. This is called multipath difference. This kind of interference affects IR, RF and MW systems. Frequency Hopping Spread Spectrum (FHSS) intuitively solves this problem by hopping to other frequencies. Other systems apply algorithms to avoid this i.e., array antennas. A subset of multipath is Rayleigh fading (24). This happens when signals are arriving from different directions and the difference in path lengths is a multiple of half the wavelength. This can cancel the signal completely, so is undesirable. IR is not effected by Rayleigh fading because the wavelengths used in IR are very small.

2.1.4 Medium Access

2.1.4.1 Medium Access Control Protocol

Most wired LANs use Carrier Sense Medium Access with Collision Detection (CSMA/CD) (25) as the MAC protocol, however in ad hoc networks we use Carrier Sense Medium Access with Collision Avoidance (CSMA/CA or MACA) (26). Carrier sense means that the station will listen before it transmits. If someone is already transmitting, sender waits and tries again later. When two stations send at the same time, transmissions collide and information will be lost. Collision detection handles this situation, by listening the signal it is transmitting to ensure no collision. Whenever a collision occur, nodes stop and try again after a delay, which is determined by the back-off algorithm. This technique becomes ineffective in wireless medium. Hidden terminal problem can occur when node A unwittingly interferes with the transmission of node C. Node C cannot hear node A, so if A is transmitting, node C will not know that A is in trans-
mission and it will decide to transmit as well. This will result in collision at B and loss of information. Carrier Sense Medium Access with Collision Avoidance (CSMA/CA or MACA) (26) solves this problem. Sender listens before it sends. If someone is already transmitting, it waits. If collision occurs, both sender execute back-off algorithm like in CSMA/CD. If no one is transmitting, it sends a short message Request To Send (RTS), which means the node is ready to send. This message contains destination address and the duration of the transmission. Other stations now know that they must wait during the next transmission. Destination node sends back Clear To Send (CTS) message. Each packet is acknowledged. When an acknowledgement is not received, MAC layer retransmits the data. This is a 4 way handshake, accepted as a standard and used (17) by 802.11.

2.2 Routing Protocols

A routing protocol is needed whenever a packet needs to be handed over to several nodes to arrive at its destination. A routing protocol has to find a route for the packet and then deliver it to correct destination. There are existing routing protocols such as distance vector and link state. However these are called non-adaptive protocols and were designed for networks with static infrastructure, and dynamic topology was not considered while they were being designed. Obviously, it will be unsuitable to use any non-adaptive protocol for a highly mobile network. Those routing protocols periodically emit control messages, which is undesirable for large network with long links. This results in large number of control messages. These are undesirable factors for mobile nodes, since they use central processing unit (CPU), engage radio transmitters and receivers even during the idle period. Consequently, these protocols can over use the resources and drain the available battery energy. All conventional routing protocols assume that routes are bi-directional and in equal quality, however in ad hoc networks, this is not
always the case. Protocols can be classified in three categories:

- Centralized or distributed,
- Adaptive or static,
- Reactive or proactive or hybrid.

When a routing protocol is centralized, all decisions are made at a center node, whereas in a distributed routing protocol, all nodes share the routing decision. An adaptive protocol may change behavior according to the network status, which can be a congestion on a link or many other possible factors. A reactive protocol takes required actions such as discovering routes when needed, besides a proactive protocol discovers the routes before they are needed. Reactive methods are called on-demand routing protocols. Since they run on demand, the control packet overhead is greatly reduced. Proactive methods keep tables of routes, and maintain those tables periodically. Hybrid methods use both approaches to find a optimal route. Zone routing protocol is an example to hybrid methods. Associativity Based Routing (ABR) is an adaptive protocol, where associativity is related to spatial, temporal and connection stability of a mobile host. There are numerous routing protocols that are proposed for MANET. In this section we focused on surveying table-driven, on-demand, and power aware protocols only, which were directly or indirectly related to this study of comparing prominent proactive and reactive ad hoc routing protocols.

2.2.1 Basics Of Routing Protocols

Distance Vector Routing. Distance-vector approaches are based on the classical Bellman-Ford algorithm. Each node maintains a list of distances from each of its neighbors to the intended destination. A node chooses the neighbor with the shortest distance to the destination as the next hop. The protocol keeps a routing table only for
its outgoing transmissions (25)(35). It only feeds the link cost information it estimates to only its neighbors by broadcasting, which is not flooding. All nodes calculate shortest path to their demanding destination using those broadcast information. This is a distributed protocol, therefore it is expected that, when we have more contributing nodes in the network, we have less stale, more accurate and shorter path estimation of routes for travelling packets. However, these protocols must incorporate a mechanism to prevent routing loops, which are a well known problem with the Bellman-Ford algorithm.

**Link State Routing.** This protocol keeps a routing table for complete topology which is built up finding shortest path of link costs (25)(35). Link cost information is periodically transmitted by all nodes using flooding technique. Each node updates its table using new link cost information gathered. Link cost information may be inconsistent because of dynamic behavior of topology in wireless medium, such as instantaneously incorrect long propagation delays. This may result in short-term long routing loops, until the next link update which causes recovery.

**Source Routing.** In this protocol, all data packets carry their routing information as their header (25). The routing decision is made at departure. Loops are avoided, however protocol overhead is expected to be cumbersome. For highly mobile topologies this is inefficient, since the protocol becomes inaccurate as result of route invalidation during the packet transmission.

### 2.3 Associativity Based Routing (ABR)

The associativity-based long-lived routing (ABR) protocol (36) for ad hoc mobile networks is a simple and bandwidth-efficient distributed routing protocol, which does not attempt to consistently maintain routing information in every node. In an ad hoc wireless network where mobile hosts are acting as routers and where routes are made
inconsistent by mobile hosts’ movement, an Associativity Based Routing scheme selects routes based on nodes having associativity states that imply periods of spatial, temporal, connection and signal stability. In this manner, the routes selected are likely to be long-lived resulting in higher attainable throughput. The proposed protocol is based on source-initiated on-demand routing. Route requests are broadcast on need basis. To shorten the route discovery time when the association property is violated, the localized-query and quick-abort mechanisms are respectively incorporated into the protocol. The association property also allows the integration of ad hoc routing into a base station oriented wireless LAN environment, providing the fault tolerance in times of base station failures. The draft describes the protocol functions and information about packet headers and routing tables.

2.3.1 Ad hoc On Demand Distance Vector Protocol (AODV)

The Ad Hoc On-Demand Distance Vector (AODV) (9) routing protocol 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, low network utilization, and determines unicast between sources and destinations. It uses destination sequence numbers to ensure loop freedom at all times, solving problems associated with classical distance vector protocols.

2.3.2 Bordercast Resolution Protocol (BRP)

The Bordercast Resolution Protocol (BRP) (42) provides the bordercasting packet delivery service used to support network querying applications. The BRP uses a map of an extended routing zone, provided by the local proactive Intrazone Routing Protocol (IARP), to construct bordercast (multicast) trees, along which query packets are directed. Within the context of the hybrid ZRP, the BRP is used to guide the route requests of the global reactive Interzone Routing Protocol (IERP). The BRP employs
special query control mechanisms to steer route requests away from areas of the network that have already been covered by the query. The combination of multicasting and zone-based query control makes bordercasting an efficient and tunable service that is more suitable than flood searching for network probing applications like route discovery.

2.3.3 Core Extraction Distributed Ad hoc Routing (CEDAR)

CEDAR (38) is a Core-Extraction Distributed Ad hoc Routing algorithm for QoS routing in ad hoc network environments. CEDAR has three key components:

1. The establishment and maintenance of a self-organizing routing infrastructure, called the “core”, for performing route computations.

2. The propagation of the link-state of stable high-bandwidth links in the core through “increase/decrease” waves.

3. A QoS route computation algorithm that is executed at the core nodes using only locally available state.

2.3.3.1 Establishment And Maintenance Of A Core Using Local Core Extraction

CEDAR does core extraction in order to extract a subset of nodes in the network that would be the only ones that perform state management and route computation. The core extraction is done dynamically by approximating a minimum dominating set of the ad hoc network using only local computation and local state. When the network topology changes, the core computation and core management, which are purely local computations, enable the core to adapt efficiently to the dynamics of the network.

Link State Propagation Using Increase/Decrease Waves While it is possible to execute ad hoc routing algorithms using only local topology information at the
core nodes, QoS routing in CEDAR is achieved by propagating, in the core, the bandwidth availability information of stable links. The basic idea is that the information about stable high-bandwidth links can be made known to core nodes far away in the network, while information about dynamic links or low bandwidth links should remain local. The key questions to answer in link state propagation are that when should an increase/decrease wave be initiated, how far should a wave propagate, and how fast should a wave propagate.

Route Computation Route computation first establishes a core path from the domain of the source to the domain of the destination. This initial phase involves probing on the core, and the resultant core path is cached for future use. The core path provides the directionality of the route from the source to the destination. Using this directional information, CEDAR iteratively tries to find a partial route from the source to the domain of the furthest possible node in the core path.

2.3.4 Cluster Based Routing Protocol (CBRP)

Cluster Based Routing Protocol (CBRP) (43) is a routing protocol designed for use in mobile ad hoc networks. The protocol divides the nodes of the ad hoc network into a number of overlapping or disjoint clusters in a distributed manner. A cluster head is elected for each cluster to maintain cluster membership information. Inter-cluster routes are discovered dynamically using the cluster membership information kept at each cluster head. By clustering nodes into groups, the protocol efficiently minimizes the flooding traffic during route discovery and speeds up this process. Furthermore, the protocol takes into consideration of the existence of uni-directional links and uses these links for both intra-cluster and inter-cluster routing.
2.3.5 Dynamic Source Routing Protocol (DSR)

The Dynamic Source Routing protocol (DSR) (45) is a simple and efficient routing protocol designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes. DSR allows the network to be completely self-organizing and self-configuring, without the need for any existing network infrastructure or administration. The protocol is composed of the two mechanisms of “Route Discovery” and “Route Maintenance”, which work together to allow nodes to discover and maintain source routes to arbitrary destinations in the ad hoc network. The use of source routing allows packet routing to be trivially loop-free, avoids the need for up-to-date routing information in the intermediate nodes through which packets are forwarded, and allows nodes forwarding or overhearing packets to cache the routing information in them for their own future use. All aspects of the protocol operate entirely on-demand, allowing the routing packet overhead of DSR to scale automatically to only that needed to react to changes in the routes currently in use. This document (45) specifies the operation of the DSR protocol for routing unicast IP packets in multi-hop wireless ad hoc networks.

2.3.6 DSR-Flow protocol

This protocol (40) is an extension to the Dynamic Source Routing protocol (DSR), a simple and efficient routing protocol designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes. DSR enables the sender of a packet to determine the sequence of nodes through which the packet must be forwarded to reach the intended destination node, and to route that packet along that sequence of hops by including a source route header in the packet. All aspects of the protocol operate entirely on-demand, allowing the routing packet overhead of DSR to scale automatically to only that needed to react to changes in the routes currently in use. The DSR extension defined in this document (40), known as “low state”, allows the routing of most packets
without an explicit source route header in the packet, further reducing the overhead of the protocol while still preserving the fundamental properties of DSR’s operation.

2.3.7 Fisheye State Routing Protocol (FSR)

The Fisheye State Routing (FSR) (41) algorithm for ad hoc networks introduces the notion of multi-level “scope” to reduce routing update overhead in large networks. A node stores the Link State for every destination in the network. It periodically broadcasts the Link State update of a destination to its neighbors with a frequency that depends on the hop distance to that destination (i.e., the “scope” relative to that destination). State updates corresponding to far away destinations are propagated with lower frequency than those for near by destinations. From state updates, nodes construct the topology map of the entire network and compute efficient routes. The route on which the packet travels becomes progressively more accurate as the packet approaches its destination. FSR resembles Link State routing in that it propagates Link State updates. However, the updates are propagated periodically as aggregates, instead of being flooded individually from each source. FSR leads to major reduction in link overhead caused by routing table updates. It enhances scalability of large, mobile ad hoc networks.

2.3.8 Host Specific Routing (HSR)

This memo overviews the need for intra-domain Host specific routing (HSR) in the Internet. Host Specific Routing (46) provides a number of benefits if route entry and look-up scalability issues can be adequately addressed. These benefits are the enabling of flat routing domains that eliminate the need for hierarchy and associated configuration, and the potential to support rapid movement of IP addresses through the routing fabric. This draft describes some of the current work in this area, including TORA and Wireless Internet Protocol (WIP), and the as yet unresolved research issues associated with large-scale host routing.
2.3.9 Intrazone Routing Protocol (IARP)

This document (47) describes the Intrazone Routing Protocol (IARP), a limited scope proactive routing protocol used to improve the performance of existing globally reactive routing protocols. With each node monitoring changes in its surrounding R-hop neighborhood (routing zone), global route discoveries to local destinations can be avoided. When a global route search is needed, the IARP’s routing zones can be used to efficiently guide route queries outwards via bordercasting, rather than blindly relaying queries from neighbor to neighbor. The proactive maintenance of routing zones also helps improve the quality of discovered routes, by making them more robust to changes in network topology. Once routes have been discovered, IARP’s routing zone offers enhanced, real-time, route maintenance. Link failures can be bypassed by multiple hop paths within the routing zone. Similarly, suboptimal route segments can be identified and traffic re-routed along shorter paths.

2.3.10 Interzone Routing Protocol (IERP)

This document (48) describes the Interzone Routing Protocol (IERP), the reactive routing component of the Zone Routing Protocol (ZRP). IERP adapts existing reactive routing protocol implementations to take advantage of the known topology of each node’s surrounding R-hop neighborhood (routing zone), provided by the Intrazone Routing Protocol (IARP). The availability of routing zone routes allows IERP to suppress route queries for local destinations. When a global route discovery is required, the routing zone based bordercast service can be used to efficiently guide route queries outward, rather than blindly relaying queries from neighbor to neighbor. Once a route has been discovered, IERP can use routing zones to automatically redirect data around failed links. Similarly, suboptimal route segments can be identified and traffic re-routed along shorter paths.
2.3.11 An Internet MANET Encapsulation Protocol (IMEP)

This memo (49) describes a multipurpose network-layer protocol, named the Internet MANET Encapsulation Protocol (IMEP), designed to support the operation of many routing algorithms, network control protocols or other Upper Layer Protocols (ULP) (where “upper” denotes *any* layer above IMEP) intended for use in Mobile Ad hoc Networks (MANET). The protocol incorporates mechanisms for supporting link status and neighbor connectivity sensing, control packet aggregation and encapsulation, one-hop neighbor broadcast (or multicast) reliability, multi-point relaying, network-layer address resolution and provides hooks for inter-router authentication procedures. Indirectly, the IMEP also puts forth a framework for MANET router and interface identification and addressing.

2.3.12 Landmark Routing Protocol for Large Scale Networks (LANMAR)

The Landmark Routing Protocol (LANMAR) (50) utilizes the concept of “landmark” for scalable routing in large, mobile ad hoc networks. It relies on the notion of group mobility, i.e., a logical group (for example a team of coworkers at a convention) moves in a coordinated fashion. The existence of such logical group can be efficiently reflected in the addressing scheme. It assumes that an IP like address is used consisting of a group ID (or subnet ID) and a host ID, i.e. \((\text{Group ID}, \text{Host ID})\). A landmark is dynamically elected in each group. The route to a landmark is propagated throughout the network using a Distance Vector mechanism. Separately, each node in the network uses a “scoped” routing algorithm (e.g., FSR) to learn about routes within a given (max number of hops) scope. To route a packet to a destination outside its scope, a node will direct the packet to the landmark corresponding to the group ID of such destination. Once the packet is within the scope of the landmark, it will typically be routed directly to the destination. Remote groups of nodes are “summarized” by the corresponding
landmarks. The solution to drifters (i.e., nodes outside of the scope of their landmark) is also handled by LANMAR. Landmark dynamic election enables LANMAR to cope with mobile environments. Thus, by using the truncated local routing table and the "summarized" landmark distance vector, LANMAR dramatically reduces routing table size and update overhead in large nets. LANMAR is well suited to provide an efficient and scalable routing solution in large, mobile, ad hoc environments in which group behavior applies and high mobility renders traditional routing schemes inefficient.

2.3.13 Lightweight Underlay Network Ad hoc Routing (LUNAR)

LUNAR (Lightweight Underlay Network Ad hoc Routing) (51) is an on-demand routing system for wireless ad hoc IP networks. It incorporates an explicit "remote state patching" (RSP) approach and is based on extending ARP to do multihop name resolution for dynamically establishing a route to the destination. LUNAR creates one or more virtual IP subnets supporting unicast as well as broadcast communications. The LUNAR system is self configuring both at the level of routing as well as the level of IP interfacing. It handles node address assignment and default gatewaying automatically, enabling forming and joining of an ad hoc network. LUNAR was designed for a simplified code base and easy extensibility of its functionality.

2.3.14 Mobile Mesh Border Discovery Protocol (MMBDP)

The Mobile Mesh Border Discovery Protocol (MMBDP) (53) enables a mobile adhoc network to utilize a fixed/wired network for dissemination of routing information and for forwarding of data. MMBDP is one protocol in a set of related Mobile Mesh protocols that also includes the Mobile Mesh Link Discovery Protocol (MMLDP) and the Mobile Mesh Routing Protocol (MMRP). Together, these protocols provide a flexible, extensible mobile adhoc networking capability.
2.3.14.1 Mobile Mesh Link Discovery Protocol (MMLDP)

The Mobile Mesh Link Discovery Protocol (MMLDP) (54) provides a media independent mechanism for discovering neighbors in a mobile adhoc network, and is capable of determining whether links are unidirectional or bidirectional.

2.3.14.2 Mobile Mesh Routing Protocol (MMRP)

The Mobile Mesh Routing Protocol (MMRP) (55) is a robust, scalable, efficient mobile adhoc routing protocol based upon the “link state” approach.

2.3.15 Multicast Zone Routing protocol (MZR)

This document (56) proposes a multicast protocol for Mobile Ad Hoc networks, called the Multicast routing protocol based on Zone Routing (MZR). MZR is a source-initiated on-demand protocol, in which a multicast delivery tree is created using a concept based on the zone routing mechanism. It belongs to the family of source-tree-based protocols, in which a delivery tree rooted at the source is created for each active multicast session. MZR does not depend on any underlying unicast protocol for a global routing substructure. The protocol’s reaction to topological changes is restricted to a node’s neighborhood and is not propagated throughout the network.

2.3.16 Optimized Link State Routing Protocol (OLSR)

This document (58) describes the Optimized Link State Routing (OLSR) protocol (58) for mobile ad hoc networks. The protocol is an optimization of the pure link state algorithm tailored to the requirements of a mobile wireless LAN. The key concept used in the protocol is that of multipoint relays (MPRs). MPRs are selected nodes which forward broadcast messages during the flooding process. This technique substantially reduces the message overhead as compared to pure flooding mechanism where every
node retransmits each message when it receives the first copy of the packet. In OLSR, information flooded in the network “through” these MPRs is also “about” the MPRs. Thus a second optimization is achieved by minimizing the “contents” of the control messages flooded in the network. Hence, as contrary to the classic link state algorithm, only a small subset of links with the neighbor nodes are declared instead of all the links. This information is then used by the OLSR protocol for route calculation. As a consequence hereof, the routes contain only the MPRs as intermediate nodes from a Source to a Destination. OLSR provides optimal routes (in terms of number of hops). The protocol is particularly suitable for large and dense networks as the technique of MPRs works well in this context.

2.3.17 Relative Distance Micro-Discovery Ad Hoc Routing (Rdmar) Protocol

This document (59) describes the Relative Distance Micro-discovery Ad Hoc Routing (RDMAR) protocol (59) for use in mobile ad hoc networks (MANETs). The protocol is highly adaptive, bandwidth-efficient and scalable. A key concept in its design is that protocol reaction to link failures is typically localized to a very small region of the network near the change. This desirable behavior is achieved through the use of a novel mechanism for route discovery, called Relative Distance Micro-discovery (RDM). The concept behind RDM is that a query flood can be localized by knowing the relative distance (RD) between two terminals. To accomplish this, every time a route search between the two terminals is triggered, an iterative algorithm calculates an estimate of their RD, given an average nodal mobility and information about the elapsed time since they last communicated and their previous RD. Based on the newly calculated RD, the query flood is then localized to a limited region of the network centered at the source node of the route discovery and with maximum propagation radius that equals to the estimated relative distance. This ability to localize query flooding into a limited area
of the network serves to increase scalability and minimizes routing overhead and overall network congestion.

2.3.18 Source Tree Adaptive Routing Protocol (STAR)

Unlike most of the other table-driven ad hoc protocols it does not use periodic messages to update its neighbors. STAR (61) is an attempt to try to create the same routing performance as the other table-driven protocols and still be equal or better on the bandwidth efficiency. To be able to do this the demand on route optimization has been put aside and the routes are allowed to be non-optimal since. This saves bandwidth but STAR depends on an underlying protocol which must reliably keep track of the neighboring nodes. This could be implemented with periodic messages, but it is not required to. In addition to this demand the link layer must provide reliable broadcasting or STAR will require this to be built into STAR with an extra routing-rule.

2.3.19 Topology Broadcast Reverse-Path Forwarding Protocol (TBRPF)

Topology Broadcast based on Reverse-Path Forwarding (TBRPF) (62) is a proactive, link-state routing protocol designed for use in mobile ad-hoc networks. TBRPF has two modes called full topology (FT) and partial topology (PT). TBRPF-FT uses the concept of reverse-path forwarding to reliably and efficiently broadcast each topology update in the reverse direction along the dynamically changing broadcast tree formed by the min-hop paths from all nodes to the source of the update. TBRPF-PT achieves a further reduction in control traffic, especially in large, dense networks, by providing each node with the state of only a relatively small subset of the network links, sufficient to compute minimum-hop paths to all other nodes. In both the FT and PT modes, a node forwards an update only if the node is not a leaf of the broadcast tree rooted at the source of the update. In addition, in the PT mode, a node forwards an update only if it results
in a change to the node's source tree. As a result, each node reports only changes to a relatively small portion of its source tree.

2.3.20 Temporally-Ordered Routing Algorithm Protocol (TORA)

This document (63) provides both a functional description and a detailed specification of the Temporally-Ordered Routing Algorithm (TORA). It is a distributed routing protocol for multihop networks. A key concept in the protocol's design is an attempt to de-couple the generation of far-reaching control message propagation from the dynamics of the network topology. The basic, underlying algorithm is neither distance-vector nor link-state rather it is a member of a class referred to as link-reversal algorithms. The protocol builds a loop-free, multipath routing structure that is used as the basis for forwarding traffic to a given destination. The protocol can simultaneously support both source-initiated, on-demand routing for some destinations and destination-initiated, proactive routing for other destinations.

2.3.21 Zone Routing Protocol (ZRP)

This document describes the Zone Routing Protocol (ZRP) framework (6), a hybrid routing framework suitable for a wide variety of mobile ad-hoc networks, especially those with large network spans and diverse mobility patterns. Each node proactively maintains routes within a local region also called the routing zone. Knowledge of the routing zone topology is leveraged by the ZRP to improve the efficiency of a globally reactive route query/reply mechanism. The proactive maintenance of routing zones also helps improve the quality of discovered routes, by making them more robust to changes in network topology. The ZRP can be configured for a particular network by proper selection of a single parameter, the routing zone radius.
CHAPTER 3. NETWORK SIMULATOR

3.1 Introduction

3.1.1 Network Simulator NS2

Network simulator ns2 is an event-driven network simulator used for networking research (70). It is a widely used and free of cost tool for simulating inter-network topologies to test and evaluate various networking protocols. The open source code is continuously being improved by various researchers. There is a substantial support and flexibility in ns2 to simulate various traffic generation patterns, routing and multicast protocols. However, other educational and commercial software (87) are also available and are used for network research. Some of the simulators which have been used in the past for network research are explained briefly in Appendix“C”. In order to study different networking issues like protocol interaction, congestion control, effect of network dynamics, scalability etc. it is necessary to simulate various scenarios that include different topology sizes, density distribution, traffic generation, membership distribution, real-time variance of membership, network dynamics etc. The NS2 scenario generator can be used to create different random scenarios for simulation. In NS2, characteristics of physical media of communication like delay, bandwidth, error rate, antennas and wireless physical interface parameters etc., can be defined. This helps in making the simulation studies as close to realistic scenarios as possible. NS2 provides the flexibility to add and experiment new protocols or ideas. Recently, much support has been added
for simulating wireless networks and interconnecting wired and wireless networks. Trace support in NS2 may be used to trace packets for wireless and wired scenarios. NS2 currently supports two mobile networking models. The basic wireless model was ported from CMU/Monarch group (88). It essentially consists of mobile nodes, which are movable and are able to transmit and receive on channels, with additional supporting features that allow simulations of multi-hop ad hoc networks and wireless LANs. Four ad hoc routing protocols that are currently supported are Destination Sequence Distance Vector (DSDV) (1), Dynamic Source Routing (DSR) (3), Temporally Ordered Routing Algorithm (TORA) (7) and Ad Hoc On-demand Distance Vector (AODV) (9).

3.1.2 Optimal NS2 Learning Strategy

3.1.2.1 NS2 Overview

The primary goal of NS2 is to support networking research and education by enabling researchers to design new protocols, study traffic patterns and do comparative studies of various protocols. It provides a collaborative environment of open source freely distributed codes and models to allow easy comparison of protocols.

NS2 is an event driven simulator for network research, which is freely available on internet. NS2 is being designed using object oriented features of $C^{++}$ with OTcl (Object Tool Command Language) as a front end interpreter. We use OTcl for network setup and configuration scripts by manipulating existing $C^{++}$ objects. However, in case we want to do anything that requires processing of each packet flow or if we want to change the behavior of an existing $C^{++}$ class in a way that weren’t anticipated then we have to work with $C^{++}$ language to write new code or modify an existing code. The simulator had two distinct requirements. On one hand we wanted to have a detailed simulation of protocol and on the other hand we wanted to frequently change the configuration parameters. The two functionalities were provided by using two different languages.
C++ is fast to run but slower to change whereas OTcl runs much slower but can be changed quickly.

NS2 translates the scenario into discrete events ordered by simulation time and processes them in sequential manner in an instant of virtual time taking an arbitrary amount of real time.

NS2 supports Wired networking, wireless networking and Satellite networking. In wired networks it supports Unicast, Multicast and Hierarchial Routing with UDP and TCP as transport and web, ftp, telnet, and cbr being the traffic sources. The queuing techniques include drop-tail, RED (Random Early Drop), FQ, SFQ, and DRR. In case of wireless networks it supports Mobile IP and Ad Hoc routing. NS2 can also be used for studying satellite network routing and handoff etc. The physical media can be point to point LAN in case of wired networks or multiple propagation model for wireless networking. The simulator components includes NAM, the Network Animator, and traffic and topology generators. The results can be analyzed by parsing the trace file as per the requirements.

3.1.2.2 Working With Ns2

The first step in this direction would be to setup correct environments for NS2 by downloading the software from http://www.isi.edu/nsnam/ns/ (Date accessed: 02 Jun 2003). The easy way to install is “all at once”. In this study we have used ns2allinone2.1b9a version, which was released on July 3, 2002. After you are done with installation the directory tree would be as shown in Figure 3.1. After installing NS2 you have to run the validation scripts as guided after the installation process which takes a while. For NS2 installation problems, bug fixes and other help you may refer to http://www.isi.edu/nsnam/ns/ns-problems.html#ns-allinone-2.1b9 (Date accessed: 02 Jun 2003).

To run a simulation we write a TCL script “my — tcl — script.tcl” and then run the
script in unix shell as "nsmy - tcl - script.tcl". In order to write TCL scripts effectively, there are numerous tutorials available on the internet. The best way is to first write very basic tcl scripts to understand normal commands for controlling loops and assigning values to variables, and then follow some of the tutorials to implement wired network with few nodes and tcp or udp traffic. Once you understand how the nodes are configured and simulated, you can proceed to study mobile nodes and Ad hoc routing protocol agents known as "rtProto". For learning NS2 in an optimized way I have included some links in Appendix ‘B’, which needs to be followed in the same sequence as explained. For further detailed study about any aspect of the simulation the inquisitive readers can refer to NS2 manual available at http://www.isi.edu/nsnam/ns/ns-documentation.html (Date accessed:02 Jun 2003).

3.1.3 Simulation Difficulties

Some of the difficulties observed while simulating mobile ad hoc networks in NS2 are explained in the following paragraphs.
3.1.3.1 Visualizing Movements Of Mobile Hosts

Protocols designed for mobile ad hoc networks must be adaptable to the highly dynamic network topologies. To validate or evaluate a protocol, it will be investigated with various scenarios with different mobile host movements. The investigation of a protocol should answer questions such as, Does the protocol correctly capture the change of network topology? Does it choose an appropriate approach according to current topology? etc. The network topology is mainly determined by the positions of mobile host, which are determined by the movement of mobile hosts. The random waypoint model (73) is the most commonly used model to generate movements for mobile hosts. A detailed description of available mobility models is given in Appendix ‘A’. According to this model, mobile hosts move independently. The speed and direction of a movement have no relation to those of the previous movement. Because of the randomness of this model, the generated movements can hardly be studied analytically. A visualization tool is needed to demonstrate the movements. This tool must be able to visualize movements generated by any model and should also be capable of visualizing wireless links if two mobile hosts are within each others communication range. The tool should be able to demonstrate the network topology at any time. NS2 itself provides a visualization component called nam, but it is not dedicated to mobile ad hoc networks. Thus it can not show the wireless links.

3.1.3.2 Analyzing Experiment Data

NS2 uses trace file to keep track of network activities at different layers. A trace file provides the most detailed information, but it has severe drawbacks.

- **File Size:** The size of a trace file can be incredibly large. For instance, a trace file for a 1000 seconds simulation of 30 mobile hosts with 30 connections can be as large as 500M. Usually, a simulation of mobile ad hoc networks is repeat several
times to compensate the randomness of scenarios. Thus, keeping all trace files requires a huge amount of storage space.

- **Comprehensive Information:** Trace files do not provide any comprehensive information. It only contains millions of records that capture the network activities. However, most of the time, researchers are only interested in comprehensive information such as throughput, delay, etc. Thus a tool is needed to analyze trace files. The desired properties of the tool include:

1. **Generating Intermediate Data:** It should be able to generate intermediate data with reasonably detailed information that can reflect the most important facts of the simulation. The size of the file should be less than 1M.

2. **Illustration:** It should be able to illustrate the results of the simulation using graphics.

3. **Compatibility:** It should be able to export data that can be understood by other generic purpose analysis software, such as Matlab, SAS, etc.

Other NS2 users are also aware of this problem. Several analysis programs have been developed. Most of them do not provide the functionality to illustrate the results. One Matlab-base program called *tracegraph* has the desired properties, but it is terribly slow. It took 30 minutes to read a 20M trace file.

### 3.1.4 Creating Random Traffic-Pattern For Wireless Scenarios.

Random traffic connections of TCP and CBR can be setup between mobile nodes using a traffic-scenario generator script. This traffic generator script is available under ns/indep-utils/cmu-scen-gen and is called cbrgen.tcl. It can be used to create CBR and TCP traffics connections between wireless mobile nodes. In order to create a traffic-connection file, we need to define the type of traffic connection (CBR or TCP), the
number of nodes and maximum number of connections to be setup between them, a random seed and incase of CBR connections, a rate whose inverse value is used to compute the interval time between the CBR packets. So the command line looks like the following:

```
"ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed seed] [-mc connections] [-rate rate]"
```

The start times for the TCP/CBR connections are randomly generated with a maximum value set at 180.0s. Tcl script cbrgen.tcl provides details of the traffic-generator implementation. To generate a CBR connection file between 10 nodes, having maximum of 8 connections, with a seed value of 1.0 and a rate of 4.0. We type at the prompt:

```
"ns cbrgen.tcl [-type cbr] [-nn 10] [-seed 1.0] [-mc 8] [-rate 4.0] cbr -10-test"
```

The output of the generator is redirected to `cbr-10-test` file, where one of the cbr connections looks like the following:

```
# 2 connecting to 3 at time 82.557023746220864
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 0.25
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 82.557023746220864 "$cbr_(0) start"
```
Thus a UDP connection is setup between node 2 and 3. Total UDP sources (chosen between nodes 0-10) and total number of connections setup is indicated as 8 with output directed to the file cbr-10-test. Similarly TCP connection files can be created using “type” as tcp. An example would be:

```
ns cbrgen.tcl [-type tcp] [-nn 25] [-seed 0.0] [-mc 8] tcp - 25 - test
```

A typical connection from tcp-25-test looks like the following:

```
# 5 connecting to 7 at time 163.0399642433226
set tcp_(1) [$ns_ create -connection TCP $node_(5) TCPSink $node_(7) 0]
(tcp_ set window _(32)
(tcp_ set packetSize_ 512
(set ftp_(1) [$tcp_(1) attach -source FTP]
$ns_ at 163.0399642433226 “ftp_(1) start”
```

### 3.1.5 Creating Node-Movements For Wireless Scenarios.

The node-movement generator is available under ns/indep-utils/cmu-scen-gen/setdest directory and consists of setdest.cc,.h and Makefile. In order to compile the setdest.cc do the following:

1. Go to ns directory and run “configure” (you probably have done that already while building ns). This creates a makefile for setdest.

2. Go to indep-utils/cmu-scen-gen/setdest. Run “make”, which first creates a stand-alone object file for ns/rng.cc (random number generator class) and then creates the executable setdest.

3. Run setdest with arguments as shown below:

Let's say we want to create a node-movement scenario consisting of 20 nodes moving with maximum speed of 10.0 m/s with an average pause between movement being 2s. We want the simulation to stop after 200s and the topology boundary is defined as 500 x 500. So our command line will look like:

```
./setdest -n 20 -p 2.0 -s 10.0 -t 200 -x 500 -y 500 scen – 20 – test
```

The output is written to stdout by default. We redirect the output to file scen-20-test. The file begins with the initial position of the nodes and goes on to define the node movements.

```
$ns_ at 2.000000000000

“$node_(0) setdest 90.441179033457 44.896095544010 1.373556960010”
```

This line from scen-20-test defines that node_(0) at time 2.0s starts to move toward destination (90.44, 44.89) at a speed of 1.37 m/s. These command lines can be used to change direction and speed of movement of mobile nodes.

Directives for GOD (General Operations Director) are also present in node-movement file. The General Operations Director (GOD) object is used to store global information about the state of the environment, network, or nodes that an omniscient observer would have, but that should not be made known to any participant in the simulation.

Currently, the god object is used only to store an array of the shortest number of hops required to reach from one node to an other. The god object does not calculate this on the fly during simulation runs, since it can be quite time consuming. The information is loaded into the god object from the movement pattern file where lines of the form

```
$ns_ at 899.642 “$god_ set-dist 23 46 2”
```
are used to load the god object with the knowledge that the shortest path between node 23 and node 46 changed to 2 hops at time 899.642.

The setdest program generates node-movement files using the random waypoint algorithm. These files already include the lines to load the god object with the appropriate information at the appropriate time.

Another program calcdest (also available in ns/indep-utils/cmu-scen-gen/setdest) can be used to annotate movement pattern files generated by other means with the lines of god information. calcdest makes several assumptions about the format of the lines in the input movement pattern file which will cause it to fail if the file is not formatted properly.

Both calcdest and setdest calculate the shortest number of hops between nodes based on the nominal radio range, ignoring any effects that might be introduced by the propagation model in an actual simulation. The nominal range is either provided as an argument to the programs, or extracted from the header on the movement pattern file.

The path length information was used by the Monarch Project to analyze the path length optimality of ad hoc network routing protocols, and so was printed out as part of the CMUTrace output for each packet.

Hence, at the end of the node movement file we find information which includes number of destination unreachable, total number of route and connectivity changes for mobile nodes and the same info for each mobile node.
CHAPTER 4. IMPLEMENTATION OF ADHOC ROUTING PROTOCOLS

4.1 Destination Sequenced Distance Vector (DSDV)

DSDV (1) is based on the idea of the classical Bellman-Ford routing algorithm (65). It belongs to the class of distance-vector, proactive protocols. This protocol prevents loop formation by associating destination based sequence numbers with route table entries. The sequence numbers provide a way for nodes to quickly determine and discard stale route information when new information arrives.

Each node maintains routing tables specifying next hops for every known destination on the network. Nodes periodically advertise their routing tables, on the order of every few seconds, so other nodes can update their topological model of the current state of the network. When a node senses a topological change, such as a link failure, it immediately broadcasts the information to the other nodes in the network. The goal is for each node to build a complete description of the current network topology, which it uses to determine the best next hop for every known destination. Nodes prefer the route with the least number of hops to reach a given destination.

DSDV routing tables associate destination addresses with next hop addresses. Next hop addresses always refer to an immediate neighbor of the node maintaining the table. No other addresses are associated with a route destination. A route table entry also stores the route metric, which is the number of hops required to reach the destination, and the destination specified sequence number for the route.
Route advertisement begins at the destination node and propagates through the entire network. Figure 4.1 illustrates the logical propagation of a route entry from a destination node to the other nodes on the network. The destination attaches a sequence number to new route advertisements. The figure shows only relevant information for destination MH₈ (Mobile Host 8). In actuality, each node would broadcast its entire route table, not just the information for MH₈.

![DSDV propagation of route information for destination MH8. Each tuple represents the destination address, sequence number, and hop count.](image)

DSDV assumes bi-directional links between all nodes. When a node receives a route table update, it modifies its own route table if the update contains newer or better routes. Nodes always prefer newer routes (i.e. those with higher sequence numbers) and will update all of its route table entries that rely on the specified destination upon receipt of such information. Any information for a destination tagged with a smaller sequence number is immediately discarded. When more than one update arrives with the same sequence number for a destination, the route with the best metric (i.e. least hops) is chosen. Whenever the node updates a route table entry, the next-hop for a given destination is set to the address of the neighboring node that transmitted the update.

Say MH₁ begins with the following route table entry, indicating MH₈ as an immediate
neighbor.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop</th>
<th>Metric</th>
<th>Sequence Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>MH₈</td>
<td>MH₈</td>
<td>1</td>
<td>402</td>
</tr>
</tbody>
</table>

Then say MH₈ moves to the position indicated in Figure 4.1 and transmits a route table update. Upon receipt of the update message originated from MH₈, MH₁ will change its route table entry to the following.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop</th>
<th>Metric</th>
<th>Sequence Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>MH₈</td>
<td>MH₂</td>
<td>4</td>
<td>404</td>
</tr>
</tbody>
</table>

This change happens solely because of the higher sequence number. The same update message (i.e. same destination and sequence number) arrives at MH₅ from MH₈ and MH₆ respectively. MH₅ chooses the message from MH₈ because it has a better metric.

When a node detects a link failure, it sets the metric of all paths in its route table with the failed next-hop to INFINITY. Each path is also assigned a new time-stamp by the discovering node. This is the only case where a node other than the destination sets the time-stamp on a route. To differentiate this case, destinations always assign even sequence numbers and other nodes always assign odd sequence numbers. The discovering node immediately broadcasts the new routing information to its neighbors.

DSDV incorporates a few optimizations in order to reduce the network overhead imposed by the protocol. It defines two types of update packet: full dump and incremental. Full dump messages carry all available routing information from the sender’s routing table. The information likely requires multiple packets. Incremental messages carry only information changed since the last full dump. They are confined to a single packet. When nodes move infrequently, incremental dumps will likely suffice and full dumps can be transmitted less frequently. When nodes move frequently, the size of an incremental update message approaches that of a full dump message. In this case, full
dumps must be scheduled more frequently in order to reduce the size of incremental updates.

Another optimization deals with the timing of re-broadcast update messages. First, updates that include newly recorded routes (i.e., those to destinations that do not already exist in the routing table) are re-broadcast immediately. Route updates with improved metrics, however, are not immediately re-broadcast. Instead, the node delays the retransmission by the average settling time for routes to the indicated destination. Settling time is a measure of the time it takes to find an optimal route to the destination. Each node keeps track of the settling time for each destination in a separate table. By delaying retransmission of these update packets, it is more likely that subsequent nodes will receive better information (if available) by the time the packet is sent. In this way, non-optimal routes tend not to be propagated throughout the entire network.

\section*{4.2 Temporally Ordered Routing Algorithm (TORA)}

TORA \cite{7}(63) is designed to work below Internet Protocol (IP) layer. It does not have properties of link state or distance-vector algorithms, but link-reversal. The protocol is adaptive, and highly scalable. It is designed to minimize reaction to topological changes. TORA control messages are localized to a very small set of nodes near the occurrence of a topological change. To achieve this, nodes maintain routing information about adjacent nodes. TORA quickly discovers multiple routes on demand. Route does not have to be optimal, but it guarantees that all routes are loop-free. TORA only does the routing job, and heavily depends on Internet MANET Encapsulation Protocol (IMEP) \cite{49}. A good analogy would be water flowing down the hill through pipes. Hill top is the source, pipes are links, and pipe connections are nodes. TORA assigns level numbers to each node down the hill. When two intermediate nodes cannot communicate, the last node raises its level higher than any of its neighbors, so that water, which
is data, flows back out of it.

Figure 4.2 Directed acyclic graph having destination D, water flow analogy

TORA has three basic functions: Route creation, route maintenance and route deletion.

Each node keeps the following values (66):

- The old unique ID of the node that defined the new reference level.
- A clock tag set to the time of the link failure, where nodes should have synchronized clocks with an external time source such as the Global Positioning System (GPS).
- A reflection indicator bit.
- A propagation ordering parameter, height.
- The current unique ID of the node itself.

The first three elements collectively represent the reference level. “A new reference level is defined each time a node loses its last downstream link due to a link failure” (66). Last two values define an offset with respect to the reference level, which were the first three values (67).
Route is created using QUERY and UPDATE packets. When a node requires a route to a destination, it sets the height of the destination to 0 and all other node’s height set to undefined, NULL. It then broadcasts a QUERY packet with ID of the destination node. Any node receiving this QUERY packet, responds with UPDATE message containing its ID only if that node’s height is a non-NULL value. Any node receiving an UPDATE packet sets its height to one more than that of the node that generated the UPDATE. A node with higher height is considered upstream and a node with lower height downstream. UPDATE packet floods until it completes the route information. This way a Directed Acyclic Graph (DAG) is constructed from the source to the destination. The propagation may end when an intermediate route knows the rest of the route. In Figure 4.3a, the route creation process is shown. Values shown in parenthesis are the reference level and height of each node, respectively. Notice that Node 4 does not propagate QUERY from Node 6 since it has already seen and propagated QUERY message. In Figure ??b, the source may have received an UPDATE both from Node 5 or Node 6 but since Node 2 gives it less height, it retains that height.

![Figure 4.3 Route creation in TORA](image)

(a) Propagation of QUERY message through network
(b) Height of each node updated as a result of UPDATE messages
The destination replies with an UPDATE message with its current height. While UPDATE message back-propagates, each node on the path sets its height to a value greater than the height of the back-transmitting neighbor, finally the largest height on the initial node.

When an intermediate node finds out that it cannot transmit data to the next hop, it increases its height to a number greater than its neighbors, and transmits an UPDATE packet. If none of its neighbor has any height, then it re-starts the algorithm from that point. If any node detects excess route or network partition, it sends a CLEAR message and removes invalid routes from the network. For each session, a separate task of TORA is run.

![Figure 4.4 TORA, when route is broken](image)

When a node moves, the DAG route is broken and route maintenance is needed to re-establish a DAG for the same destination. When the last downstream link of a node fails, it generates a new reference level. This results in the propagation of that reference level by neighboring nodes as shown in Figure 4.4. Here, after the link failure between Nodes 3 and 4 happens, the new reference level is set as Node 4. Links are reversed to reflect the change in adapting to the new reference level. This has the same effect as reversing the direction of one or more links when a node has no downstream links. In
route maintenance phase, TORA floods a broadcast clear packet (CLR) to the network to erase invalid routes.

In TORA there is a potential for oscillations to occur, especially when multiple sets of coordinating nodes are concurrently detecting partitions, erasing nodes, and building new routes based on each other. Since TORA uses inter-node coordination, its instability problem is similar to the count-to-infinity problem in distance vector routing protocols, except that such oscillations are temporary and route convergence will surely occur. Overhead of TORA is quite high due to its nature and running on top of IMEP (49).

4.3 Ad Hoc On-Demand Distance Vector (AODV)

AODV (9) uses mechanisms of route discovery and maintenance of DSDV (1), sequence numbers and beaconing of DSR. AODV minimizes the number of broadcasts by creating routes on demand as opposed to DSDV that maintains the list of all the routes. As in Figure 4.5a, when a node requires a path to the destination, it broadcasts RREQ message to its neighbors which includes latest known sequence number for that destination. This message is flooded until information required is complete by any means. Each node receiving the message creates a reverse route to the source. As seen in Figure 4.5b, the destination sends back ROUTE REPLY message which includes number of hops in between and its sequence number. Each node receiving the reply message creates a forward route to the destination. Thus each node remembers only the next hop required to reach any of the hosts, not the whole route.

This method requires periodic updates of routing information, AODV broadcasts periodic HELLO messages. Failure to receive consecutive number of HELLO messages determines failure of a node. Whenever this is detected, source is notified with UNSOLICITED ROUTE REPLY message which includes infinite metric for the destination. In that case, the source initiates a new route discovery. This protocol is loop free, and
link breakages results in notification of demanding nodes immediately. AODV supports multicasting, and avoids Bellman Ford (65) counting to infinity problem.

AODV keeps track of the following information:

- Destination IP address.
- Destination sequence number.
- Hop count: How much hop a packet has traversed.
- Next hop: Next to be forwarded host.
- Lifetime: Duration which this route is considered to be valid.
- Active neighbor list: Neighbors which use this route entry.
- Request buffer: A request should only be processed once.

This protocol reduces the protocol overhead greatly. Triggered updates are done whenever a link failure is detected. However, notifying back gets costly whenever reply
back route is very long. AODV needs HELLO messages, but does this using IP, which is without using the link layer. Due to this, its performance is lower. HELLO messages add significant overhead to the protocol. Links are always considered as bidirectional, RREP messages are bounced back where they are originated. Bidirectional assumption might cause improper execution of the protocol.

4.4 SIMULATION MODEL

We used NS2 (70) simulator, in order to comparatively evaluate the performance of different ad hoc routing protocols. In this section, we will introduce the simulator and describe our simulation methodology. The simulation methodology can be best described by the following Figure 4.6:
4.4.1 Network Simulator

NS (70) is a discrete event simulator developed for networking research. It provides support for wired and wireless networking with multicast capabilities and satellite Networks. It has some limitations, such as 2D terrain with two way ground reflection model is used. The simulator is written in C++, accompanying OTCL script language based on Tcl/Tk. The researcher defines the network components such as nodes, links, protocols and traffic requirements using OTCL scripts. The simulator uses this script and outputs the trace at different selective layers. We used this output to calculate delays, throughput, power consumption and other performance measures.

We have used NS2 to simulate ad hoc routing protocols. The simulations were run on the latest available version of NS2 (ns-2.1b9).

The wireless model of NS2 consists of a mobile node at the core. It has additional features allowing to simulate multihop ad hoc networks, wireless LANs. A Mobile node has also the ability to move within a given topology, ability to transmit and receive signals to and from a wireless channel.

4.4.2 Physical Layer Model

Propagation models are used to determine if the data transmitted through the air has been successfully received. These models consider propagation delays, carrier sensing, and capture effects.

In NS2, for power attenuation of a signal, $1/r^2$ free space model for short distances \((r)\), $1/r^4$ ground reflection model for long distances above 100 meters are usually used which is suitable for low-gain antennas, located 1.5 meters above the ground, operating in the 1-2 GHz band (71). When a packet transmission is requested, the sender object computes the propagation delay from the sender to every other interface on the channel and schedules the packet reception event for each node.
MAC 802.11 Implementation. IEEE 802.11 MAC (17) is implemented within NS2. MAC layer handles collision detection, fragmentation and acknowledgements. The protocol may also be used to detect transmission errors. 802.11 is a CSMA/CA protocol, it avoids collisions by checking the channel before using it. If the channel is free, it can start sending, if it is not, it waits a random amount of time before re-sending. For each retry, exponential backoff algorithm is used. In a wireless medium, it cannot be assumed that all stations hear each other. If a station seizes the medium as available, it may not necessarily be so.

This problem is known as the hidden terminal problem as shown in Figure 4.7 and to overcome these problems, the collision avoidance mechanism and positive acknowledgement scheme is used together. Positive acknowledgement requires peers to retransmit data and acknowledge to each other until both are successful.

![Hidden terminal problem](image)

Figure 4.7 Hidden terminal problem that occurs when node A unwittingly interferes with the transmission of node C

In NS2, ARP (72) is implemented. It translates IP addresses to hardware MAC address before the packets sent down to MAC. The antenna gain and receiver sensibility parameters are available in NS2. There are different antennas available for simulations. The channel implementation is based on a shared media model. All mobile nodes have one or more network interfaces that are connected to a channel. A channel has a particular radio frequency with a particular modulation and a coding scheme. Channels are orthogonal, that is, packets sent on one channel do not interfere with transmission and reception in adjacent or any other channels. A packet is received if the transmission
range is within the radio propagation model calculation, and if bit errors are within tolerance limits.

**Mobile Node** This is the basic object with added functionalities, which can make movements and receive/transmit on a channel. Mobility features include node movement, periodic position updates, maintenance of topology boundary. These are implemented in C++. Creation and attachment of network components like classifiers, de-multiplexer, link layer (LL), MAC, channel within mobile node have been implemented in OTCL.

Each mobile node is attached a routing agent for calculating routes to other nodes in the network. Packets sent from the application are received by the routing agent in Figure 4.8. The agent determines a routing path for the packet and stamps it. It sends the packet down to the link layer. The link layer uses ARP to determine the hardware addresses of neighboring nodes and maps IP addresses to their correct interfaces. The packet is then sent to the interface queue, and stays there until a signal from MAC is received. It leaves the interface queue (IFQ) and waits for MAC to send it when the channel is available. The packet is copied to all interfaces at the time at which the first bit of the packet would begin arriving at the interface in a real physical system. Each network interface stamps the packet with its own properties, and invokes the propagation model. Note that the propagation model is invoked at the received part. The propagation model uses transmit and receive stamps to determine the power that the interface will receive the packet. The receiving network interface is left to decide whether the packet is received successfully or not. If successful, the packet is passed to MAC layer. If MAC layer receives this packet as error-free and collision-free, it passes the packet to node's entry point. The packet then reaches a de-multiplexer, which decides whether the packet should be forwarded again or if it has reached its destination node. If arrived point is the destination node, the packet is sent to the de-multiplexer, that
decides the application to which it should be delivered. If the packet is forwarded, this operation is repeated.

![Diagram of a mobile node in NS2, featuring DSDV agent]

---

**4.4.3 Simulation Overview**

Figure 4.9 shows the system we used to do our simulations. We generated different mobility patterns which use the waypoint model (73) (71). In the waypoint model, nodes are randomly placed in a bounded region. Each node moves to a given destination within the region boundaries with a given speed and pauses for a given duration. After the pause, the node chooses another destination and speed and continues to behave similarly. Parameters of the waypoint algorithm, except the pause time are drawn from the random
number generator for a specified distribution. For this, we used setdest utility provided with NS2. We also used setdest to obtain different mobility patterns for different seeds and generated traffic patterns for different loads.

![Diagram](image)

Figure 4.9 Simulation process

4.5 Performance Evaluation Criteria

Three important performance metrics are used to evaluate these protocols:

Packet delivery fraction  The ratio of the data packets delivered to the destinations to those generated by the CBR sources.

Average end-to-end delay of data packets  This includes all possible delays caused by buffering during route discovery latency, queuing at the interface queue, retransmission delays at the MAC, and propagation and transfer times.
Normalized routing load  The number of routing packets transmitted per data packet delivered at the destination. Each hop-wise transmission of a routing packet is counted as one transmission.

The first two metrics are the most important for best-effort traffic. The routing load metric evaluates the efficiency of the routing protocol. However, these metrics are not completely independent. For example, lower packet delivery fraction means that the delay metric is evaluated with fewer samples. In the conventional wisdom, the longer the path lengths, the higher the probability of a packet drops. Thus, with a lower delivery fraction, samples are usually biased in favor of smaller path lengths and thus have less delay.

4.6 Parsing and Analyzing Data

After each simulation, trace files recording the traffic and node movements are generated. These files need to be parsed in order to extract the information needed to measure the performance metrics. The new trace format was used for parsing.

The new trace format looks like:

```
s -t 0.267662078 -Hs 0 -Hd 1 -Ni 0 -Nx 5.00 -Ny 2.00 -Nz 0.00 -Ne
-1.000000 -Ni RTR -Nw —-Ma 0 -Md 0 -Ms 0 -Mt 0 -Ii 20 -Is 0.255
-Id -1.255 -It
```

Here, we see that a packet was sent (s) at time (t) 0.267662078 sec, from source node (Hs) 0 to destination node (Hd) 1. The source node id (Ni) is 0, it’s x-co-ordinate (Nx) is 5.00, it’s y-co-ordinate (Ny) is 2.00, it’s z-co-ordinate (Nz) is 0.00, it’s energy level (Ne) is 1.000000, the trace level (Ni) is RTR and the node event (Nw) is blank. The MAC level information is given by duration (Ma) 0, destination Ethernet address (Md) 0, the source Ethernet address (Ms) is 0 and Ethernet type (Mt) is 0.
The IP packet level information like packet id (Ii), source address.source port number is given by (Is) while the destination address.destination port number is (Id).

**Evaluating Packet delivery fraction (pdf).** Calculate the number of “sent packets” that have the trace form:

```
/s*- Nl AGT.*-Is (%d{1,3})\d{1,3} -Id (%d{1,3})\d{1,3}.*-It
cbr.*-Ii (%d{1,6})/
```

AGT $\implies$ Agent Level Trace

Calculate the number of “received packets” of the trace form:

```
/r -t (%d{1,3})\d{9}.*-Nl AGT.*-Is (%d{1,3})\d{1,3} -Id (%d{1,3})\d{1,3}.*-It cbr.*-Ii (%d{1,6})/
```

Packet delivery fraction (pdf %) = (received packets / sent packets) * 100

**Evaluating Average End-End Packet Delivery Time.** For each packet with id (Ii) of trace level (AGT) and type (cbr), calculate the send (s) time (t) and the receive (r) time (t) and average it.

**Evaluating Normalized Routing Load.** Calculate the routing packet sent:

```
/[s|f]*-Nl RTR.*-It (?=AODV|DSR|message) -Il (%d{1,4})/
```

f $\implies$ forward

RTR $\implies$ Routing Level Trace

Normalized routing load = (routing packets sent) / receives.
4.7 Performance Evaluation

The simulations were carried out with 50 nodes and a number of traffic sources/connections as 4/5, 7/10, 10/15, and 17/25. The pause time was varied as 0 (high mobility), 10, 20, 40, 100 (no mobility) and the packets were sent at a rate of 4 packets/sec. The simulations were done for DSDV, AODV and TORA protocols to compare their performance as per the established metrics. The results obtained are presented in the following graphs.

![Packet Delivery Fraction](image)

Figure 4.10 Packet Delivery Fraction(a) - 10 Sources

First, an attempt was made to compare all the 3 protocols under the same simulation environment. For all the simulations, the same movement models were used, the number of traffic sources were fixed at 5, 10, 15 and 25, whereas the maximum speed of the nodes was set to 20m/s and the pause time was varied as 0s, 10s, 20s, 40s and 100s.

Figures 4.10, 4.11, 4.12 and 4.13 highlight the relative performance of the three routing protocols. All of the protocols deliver a greater percentage of the originated data packets when there is little node mobility (i.e., at large pause time), converging to 100% delivery when there is no node motion.
Packet Delivery Comparison  The On-demand protocols, TORA and AODV performed particularly well, delivering over 98% of the data packets regardless of mobility rate as shown in Figures 4.10, 4.12 and 4.13. However, TORA performed extremely poor in case when we increased the number of sources to 25 as shown in Figure 4.11. With 25 sources TORA’s average packet delivery ratio drops to 20% at pause time 10 and 100. It essentially failed to converge in this scenario due to excessive congestion which is further explained in the subsequent paragraph.

Average End-End Packet Delivery  Theoretically, table driven protocols must exhibit lower end-to-end delay for packet delivery due to the availability of the route caches at each node unless the route has gone stale. In such cases it triggers “Route Discovery” through flooding. This may cause excessive overhead in specific cases. However, On-Demand routing protocols are expected to have large overhead due to ‘Route Discovery On-Demand”.

The results exhibits generally comparable performance of DSDV and AODV as per Figures 4.14, 4.15, 4.16 and 4.17, which largely outperformed the TORA’s perfor-
performance. The worst performance of TORA is seen to be in Figure 4.14 which is Average End to End Delay for packet delivery of 0.14 sec with 25 sources at pause time of 10 sec. This behavior supplements the findings made in the above paragraph, which is attributed to the large routing packets over-head causing severe congestion and hence enhanced delay for end-to-end packet delivery.

**Normalized Routing Load Comparison** In all cases as per Figure 4.18, 4.19, 4.20 and 4.21, DSDV and AODV demonstrates significantly lower routing load than TORA, with the factor increasing with a growing number of sources. The worse performance is again in case of 25 sources as shown in Figure 4.18.

The detailed investigated of this particular observation revealed that TORAs overhead is the sum of a constant mobility-independent overhead and a variable mobility-dependent overhead. The constant overhead is the result of IMEPs neighbor discovery mechanism, which requires each node to transmit at least 1 HELLO packet per BEACON period (1 second). For 100-second simulations with 50 nodes, this results in a minimum overhead of 5,000 packets. The variable part of the overhead consists of the routing packets TORA uses to create and maintain routes, multiplied by the number of retransmission and acknowledgment packets IMEP uses to ensure their reliable, in-order delivery.

In many of our scenarios with 25 sources, TORA essentially underwent congestive collapse. A positive feedback loop developed in TORA/IMEP wherein the number of routing packets sent caused numerous MAC-layer collisions, which in turn caused data, ACK, and HELLO packets to be lost. The loss of these packets caused IMEP to erroneously believe that links to its neighbors were breaking, even in pause time 100 scenarios when all nodes were stationary. TORA reacted to the perceived link breakages by sending more UPDATEs, which closed the feedback loop by directly causing more congestion. More importantly, each UPDATE sent required reliable delivery, which
increased the system's exposure to additional erroneous link failure detections, since the failure to receive an ACK from retransmitted UPDATES was treated as a link failure indication.
Figure 4.12  Packet Delivery Fraction(c) - 5 Sources

Figure 4.13  Packet Delivery Fraction(d) - 15 Sources
Figure 4.14 Average End to End Delay(a) - 10 Sources

Figure 4.15 Average End to End Delay(b) - 25 Sources
Figure 4.16  Average End to End Delay(c) - 5 Sources

Figure 4.17  Average End to End Delay(db) - 15 Sources
Figure 4.18 Normalized Routing Load(a) - 10 Sources

Figure 4.19 Normalized Routing Load(b) - 25 Sources
Figure 4.20  Normalized Routing Load(c) - 5 Sources

Figure 4.21  Normalized Routing Load(d) - 15 Sources
CHAPTER 5. CONCLUSION AND FUTURE WORKS

5.1 Conclusions

It is not simple to determine which of the three protocols under comparison is the best for ad hoc networks. No protocol is ideal for all scenarios. A good criterion to choose a protocol might be the size and expected traffic load in the target network. Table-driven protocols like DSDV face scaling problems, as the number of nodes in the network grows much larger than the network size considered in this study, because the overhead traffic grows linearly with the number of destinations. On-demand routing protocols face scaling problems when the number of nodes is large and each such node has a good likelihood of contacting several other nodes in the network, because the overhead grows linearly with the number of active destinations.

According to our simulation study, in small networks (50 nodes or so), AODV delivers more packets than the other two protocols, while DSDV incurs less overhead than the other protocols. DSDV always has the lowest latency, which is interesting from the fact that routes exist without the need for on-line searches of such routes. The control overhead of DSDV does not change much when the number of data sources changes. The control overhead for TORA increases substantially when the number of data sources increases as compare to AODV and DSDV.

Each of the protocols studied performs well in some cases yet has certain drawbacks in others. DSDV performed quite predictably, delivering virtually all data packets when node mobility rate and movement speed are low. TORA, although the worst performer
in our experiments in terms of routing packet overhead, still delivered over 80% of the packets in scenarios with 5, 10 or 15 sources. At 25 sources, the network was unable to handle all of the traffic generated by the routing protocol and a significant fraction of data packets were dropped. Finally, AODV performs almost as well as DSDV at all mobility rates and movement speeds and accomplishes its goal of eliminating source routing overhead, but it still requires the transmission of many routing overhead packets and at high rates of node mobility is actually more expensive than DSDV.

Varying the degree of mobility, or the moving speed of each node in the network, is a useful way to test how adjustable a routing protocol is to the dynamic environment. There have been several mobility models used in the past. We chose the “random waypoint” model because this has been used more widely than other mobility models. The details are provided in “Appendix-A”.

We found similarities in the results from prior simulation studies using ns-2 as well as differences. This indicates that the simulation results serve as a good reference for studying protocol features and for comparing different protocols, but are not accurate enough for deriving conclusions about the expected performance of a given protocol in a real network.

In many scenarios simulated in previous simulation studies of ad-hoc networks, nodes were usually densely connected. In a highly dense network, almost every node has at least a path to any other node, usually just a few hops away. Meanwhile due to the high volume of routing control messages, congestion happens frequently in such networks. A sparsely connected ad hoc network bears different characteristics. In such a network, paths between two nodes do not always exist, and routing choices are more obviously affected by the mobility of the network.
5.2 Future Work

In future, extensive complex simulations can be carried out on high performance workstations by varying size of network, number of nodes in a given space, increasing simulation time and varying number of sources, in order to gain a more in-depth performance analysis of the ad hoc routing protocols. The comparative study can also include simulating DSR and ZRP for further in-depth analysis and performance evaluation in densely and sparsely connected network scenarios. DSR and AODV are presently considered to be the favorites for standardization, but both protocols suffers scaling problem. Another step further can be to modify these protocols to overcome these shortcomings. The proposed solutions can be GPS aided routing mechanism, and restricting flow of route request packets in a certain area based on a pre-established criteria and information acquired from GPS attached to each node.
APPENDIX A. MOBILITY MODELS USED IN SIMULATIONS

A.1 Random Waypoint Model (RWM)

Johnson and Maltz describe in (77) the Random waypoint model. It works as follows. All nodes are uniformly distributed around the simulation area at starting time. Each node then choose a random destination and moves there with a speed uniformly distributed over \([0, vmax]\). Then there is a pause time which could be selected be '0' to give continuous motion.

A.2 Random Direction Model (RDM)

In (78) Royer et al describe another random based model. This a more “stable” model than a random waypoint model. At start the nodes selects a random direction and starts to move along it. Since the area of simulation is confined the node may end up reaching one of the boundaries during the simulation. When a boundary is reached the node pause for a given time and then chooses a new direction to travel. Since the node is on a boundary the selectable angle is 180 degrees. The result of this model is a more stable distribution of the nodes than the RWM. The behavior can be thought as a micro-cell of a larger area which is a useful property.
A.3 Modified Random Direction Model (MRDM)

A second more advanced version described by Royer et al in (78). To give a even more realistic simulation the Random Direction Model was extended with a extra choice for the nodes when their pause time is over. The nodes don’t have to travel all the way to the boundary but could stop anywhere along the path.

A.4 Brownian Model (BM)

Hu and Johnson describe in (79) another way of modelling the speed of the nodes. Changes speed and direction at discrete time intervals and at the beginning of each interval each node chooses \( r \in [0, v_{max}] \) and \( \theta \in [-\pi, \pi] \) moves with velocity vector \( (r \sin \theta, r \cos \theta) \). This model is very similar to the random direction model except for the speed which is smooth in this model.

A.5 Column Model (CM)

A mobility model suited for experiments is described by Sanchez in (80). Nodes are only moving along the x-axis. The initial position of node \( i \) is \( (10i, 10i) \) and the node changes the speed \( v \in [0, v_{max}] \) at the discrete intervals. This will produce a mobility pattern that is one dimension simpler than the random mobility model since the nodes only move along the x-axis.

A.6 Random Gauss-Markov Model (RGM)

RGM uses discrete time intervals to divide up the motion. The nodes update their velocity vectors at the beginning of each interval according to:

\[
v_{xt} = \alpha_{xt-1} + (1 - \alpha) \cdot \bar{v}_x + R\sqrt{1 - \alpha^2}
\]  

(A.1)
\[ v_{yr} = \alpha_{yr-1} + (1 - \alpha) \cdot \bar{v}_y + R\sqrt{1 - \alpha^2} \]  \hspace{1cm} (A.2)

\( R \) is a random variable with mean 0 and variance \( \sigma \). This model is described by Sanchez (80) and further developed by Liang and Haas (81).

### A.7 Pursue Model (PM)

Another model done by Sanchez (82) in order to try to create group movement. One node in each group is moving according to the random waypoint model. The rest of the group is moving towards the target that the “leading” is aiming for. The speed of the pursuing nodes is chosen uniform random in the range \([v_{pmin}, v_{pmax}]\).

### A.8 Exponential Correlated Random Model (ECR)

The ECR is able to model all possible movements of individuals and groups. This is done by changing the parameters of a motion function. A new position \( b(t - 1) \) is a function of the previous position \( b \) to which a random deviation is added. The function \( b(t) = (r_t, 0_t) \) can be defined either for a single node or a group at time \( t \). \( r \) is a random Gaussian variable with variance \( \sigma \). The parameters are then changed to give different mobility patterns. Very hard to create a predefined motion pattern by selecting the parameters. This model is described by BBN in (83).

### A.9 Reference Point Group Mobility Model (RPGM)

Ho et al describes another way to simulate group behavior in (84) where each node belong to a group where every node follow a logical center reference point. The nodes in a group are usually randomly distributed around the reference point. The different nodes use their own mobility model and is then added to the reference point which drives
them in the direction of the group. This general description of group mobility can be used to create a variety of models for different kinds of mobility applications.

A.10 Individual Simulated Behavioral Model (ISB)

This is another new and different idea how to do more accurate and better simulations. They use a theory about an individually simulated behavioral model where all objects have their own properties. They verified their idea with DSR and proved that it generates reproducible and “realistic” mobility patterns (85).
APPENDIX B. LEARNING NS-2

The basic overview and general introduction to NS-2 is available in Chapter 3. However, to learn NS-2 in an optimized way you can follow the links given below. This is just a recommendation as I feel it to be the most optimized and appropriate learning strategy. The readers are free to adopt another approach or methodology to become proficient in NS-2. The learning sequence given below ensures that the users can write simple to moderately complex NS-2 simulations, however, for adding new functionality in an existing routing protocol or adding new protocol need further in depth study and a good knowledge of C++.

- NS main web page guides you how to download, install and verify the installation of NS-2. I recommend to install allinone ns-2.1b9, which I found relatively bug free for Ad Hoc network simulations.
  
  http://www.isi.edu/nsnam/ns (Date accessed: 02 Jun 2003)

- NS installation problems and bug fixes and help.
  
  http://www.isi.edu/nsnam/ns/ns-problems.html#ns-allinone-2.1b9 (Date accessed: 02 Jun 2003)

- Marc Greis Tutorial maintained by VINT group.
  

- NS by example authored by ‘Jae Chung and Mark Claypool’.
  
  http://nile.wpi.edu/NS/ (Date accessed: 02 Jun 2003)
• Simulating Ad-hoc routing protocols. A power point presentation by 'Mahesh'. The presentation explains simulating Ad-hoc networks with small example scripts.


• NS-2 Scenario Generation.

http://www.isi.edu/nsnam/ns/ns-scengeneration.html (Date accessed: 02 Jun 2003)

• NS Tutorial which takes you from basic Wired network simulations to large Wireless simulations.


• NS-2 Running Wireless Simulation in NS-2.


• NS2 Tutorial in power point presentation form by 'Sung Park' of Network and Embedded Systems Lab (NSEL), UCLA. This presentation gives you good insight about simulating Ad-Hoc networks using ns-2.1b9a.

http://nesl.ee.ucla.edu/courses/ee206a/2002s/guest_presentations/GP02_Park_ns2.ppt (Date accessed: 02 Jun 2003)

• NS Manual. A detailed treatment to NS2 functionalities.


You must remember that there is no short cut to “learning by doing”. No crash course on TCL or on NS2 and no tutorial would be helpful unless you do practice and learn by experience. You are also encouraged to find other relevant material on the Web as you feel necessary to supplement your learning desire.
APPENDIX C. OTHER NETWORK SIMULATORS

A detailed account of various Simulators is available on the web (87). The list categorizes them into educational/government Simulators and Commercial simulators. The available links can take inquisitive readers to the relevant documents giving a detail account of each simulator. Some of the well researched and known simulators are briefly explained in subsequent sections.

C.1 GloMoSim

GloMoSim (86)provides a scalable simulation environment for wireless and wired network systems. It is being designed using the parallel discrete-event simulation capability provided by Parsec. GloMoSim currently supports protocols for a purely wireless network which includes AODV, DSR, Fishye, LAR, ODMRP and WRP. It also anticipate adding functionality to simulate a wired as well as a hybrid network with both wired and wireless capabilities. Most network systems are currently built using a layered approach that is similar to the OSI seven layer network architecture. GloMoSim uses a similar layered approach. Standard APIs are used between the different simulation layers. It allows rapid integration of models developed at different layers by different people.

To run GloMoSim, you will require the latest Parsec compiler which is now included with the GloMoSim distribution. If you want to develop your protocols in GloMoSim, you should have some familiarity with Parsec, but you don’t need to be an expert. Most protocol developers will be writing purely C code with some Parsec functions for time
management. Hence, you will need to use the Parsec compiler. Parsec code is used extensively in the GloMoSim kernel. Most users do not need to know how the kernel works. If you are interested in the GloMoSim kernel, you will need to have extensive knowledge about Parsec.

C.2 QualNet

QualNet is a commercial GloMoSim Based Product. QualNet model libraries and integration tools provide network designers with a large library of networking options. Collectively, the libraries simulate a diverse set of networking applications. The QualNet model library consists of 'Standard Library' (included with base QualNet product), 'MANET Library', 'QoS Library', 'Satellite Library' and 'Cellular Library'. The latest release of QualNet 3.6 was released on April 1, 2003 which includes libraries of Cellular GSM, Detailed Satellite, and IEEE 802.11a. In addition to nine new models added to the QualNet library, QualNet Developer 3.6 includes a number of functional improvements. QualNet Developers code editor and assisted code development tools have been greatly enhanced, and threaded traffic file formats are now supported. The product is also available with concessions for educational purposes to the universities and students.

C.3 OPNET

OPNET is a commercial software which supports Windows and Unix environments and provides a Graphical User Interface. The OPNET Wireless module extends the functionality of OPNET's core products with high-fidelity modelling, simulation, and analysis of wireless networks, including the RF environment, interference, transmitter/receiver characteristics, full protocol stack including MAC, routing, higher layer protocols, and applications, node mobility, and interconnection with wire-line transport networks.
Researchers can take advantage of OPNET’s wireless simulation capabilities and rich protocol model suites to optimize their R&D efforts. Wireless network managers, architects, and operations professionals can analyze end-to-end behavior, tune network performance, and evaluate growth scenarios for revenue-generating network services. It uses parallel simulations which is now a standard feature of the Wireless module to leverage the power of multiple processors for wireless link computations. The Wireless module provides the industry’s most flexible and scalable wireless network modelling environment, and includes a broad range of powerful technologies for accelerating simulation executions.
BIBLIOGRAPHY


(Date accessed : 02 Jun 2003)

(Date accessed : 02 Jun 2003)


[23] Lucent Technologies, Solutions and Products.  


[38] Raghupathy Sivakumar, Prasun Sinha, Vaduvur Bharghavan, Core Extraction Distributed Ad hoc Routing (CEDAR) Specification, IEEE Journal on Selected Areas In Communication, VOL 17, NO 8, August 1999.


http://www.docs.uu.se/docs/research/projects/selnet/lunar/. (Date accessed : 06 Jun 2003)


[64] The Official IETF Working Group Manet webpage,


http://www.isi.edu/nsnam/ns/ (Date accessed: 06 Jun 2003)


[84] Xiaoyan Hong, Mario Gerla, Guangyu Pei, Ching-Chuan Chiang A, *Group Mobility Model for Ad Hoc Wireless Networks*, University of California.


[87] Educational And Commercial Softwares For Network Simulation.

[88] Rice University Monarch Project.
ACKNOWLEDGEMENTS

I sincerely extend my gratitude and thanks to my advisor Dr Mani Mina and my co-advisor Dr Doug Jacobson. Their knowledge of the subject with excellent abilities as a teacher had been a source of inspiration. The accommodating, composed and patient nature of my advisors always made it simple to approach them with any kind of problems. Dr Mani Mina’s immense encouragement and valuable discussions always helped me to remain focused and are commendable.

I am also thankful to Dr Thomas. J. Rudolphi, for sparing his valuable time to serve on my committee.

Before and during the completion of this research work, I benefited from many enlightening discussions with colleagues and friends. It would be difficult to list them all here. At the risk of forgetting some of the most significant contributors, I would like to extend special thanks to Al-Karaki Jamal Nazzal.

Last but not the least, I would like to thank my dear parents and appreciate the support and encouragement extended to me by my wife Zahra to pursue my graduate studies. The best wishes extended by my kids Husnain, Saqlain, and Aainy gave right perspective to my studies and future goals. I owe all of them a great deal.