Routing Protocols for Ad Hoc Wireless Networks

By

Mohammad Ismail

A graduate project submitted to the Faculty of Graduate Studies in partial fulfillment of the requirements for the degree of

Master of Science
In
Information and System Science

Department of Mathematics and Statistics
Carleton University
Ottawa, Ontario
Canada, K1S 5B6

August, 2001

Supervisor: Professor Thomas Kunz

Acknowledgements

I would first like to thank my supervisor, Professor Thomas Kunz, Department of Systems and Computer Engineering, for his guidance during my whole project works. I also give extra special thanks to him for dedicating his valuable time whenever I needed to discuss project related works without any delay.

I would also like to thank Professor Sam Malkonian, Director, Ottawa - Carleton Institute of Mathematics and Statistics for giving permission to pursue my project work in my interested field "Routing in Ad Hoc Wireless Networks."

I give thanks to Diane Smyth, graduate secretary, Department of Mathematics and Statistics for organizing the project talk. I give extra thanks to previous graduate secretary of the same department, Pat Lamoureux for her always being willing to help find solutions to any problems I had since I admitted to the program.

I thank Lal Sebastian, computer systems manager for keeping lab up and running as well as helping me as and when needed.

Abstract

An ad hoc network is a collection of inter-connected, arbitrarily arranged mobile hosts and routers, with a topology that changes unpredictably and relies on no fixed infrastructure or central coordinating entity. Nodes communicate directly between one another over wireless channels. Because the transmission range of these nodes is limited, a routing protocol is needed to enable communication between them. However, because of the portable nature of the wireless devices and the wireless transmission medium, ad hoc network have many characteristics that render routing protocols designed for wired networks inapplicable. The primary goal of an ad hoc network routing protocol is correct and efficient route establishment between a pair of nodes so that messages may be delivered in a timely manner. Route construction should be done with a minimum of overhead and bandwidth consumption.

This report examines twelve unicast and four multicast routing protocols for ad hoc networks by presenting their characteristics and functionality, and then provides a comparison, performance evaluation and discussion of their respective strengths and weaknesses. The report concludes with some observations on the open areas for further investigation and suggestions to address some of these issues.

Table of Contents

1 Introduction 1
1.1 Motivation 1
1.2 Infrastructured Wireless Network 1
1.3 Infrastructureless Wireless Networks 2
1.4 Applications 2
1.5 Characteristics 3
1.6 Desirable properties 3
1.7 Description 4

2 Routing 5
2.1 Conventional protocols 5
2.2 Link State 5
2.3 Distance Vector 5
2.4 Source Routing 5
2.5 Flooding 6
2.6 Classification 6

3 Existing Unicast routing Protocols 7
3.1 Proactive (Table - driven) Protocols 7
3.1.1 Destination Sequenced Distance Vector (DSDV) 8
3.1.1.1 Protocol Overview 8
3.1.1.2 Route Advertisements 8
3.1.1.3 Route Table Entry and Route Selection Criteria 8
3.1.1.4 Responding to Topology Changes 9
3.1.1.5 Extending Base Station Coverage 9
3.1.1.6 Examples of DSDV in Operation 9
3.1.1.6.1 Damping Fluctuations 11
3.1.1.7 Properties of the DSDV Protocol 11
3.1.1.7.1 Proof of loop-free property 11
3.1.1.8 Comparison with Other Methods 12
3.1.1.9 Strengths and Weaknesses 12
3.1.2 Clusterhead Gateway Switch Routing Protocol 13
3.1.2.1 Cluster Formation and Election of Cluster heads 13
3.1.2.2 Gateways 13
3.1.2.3 Clustering 13
3.1.2.4 Routing 14
3.1.2.5 Strengths and Weaknesses 14
3.1.3 The Wireless Routing Protocols 15
3.1.3.1 Protocol Overview 15
3.1.3.2 Information Maintained at Each Node 15
3.1.3.3 Information Exchanged among Node 15
3.1.3.4 Routing Table Updating 16

3.1.3.4.1 Handling Topology and Link-Cost Changes 16
3.1.3.5 Examples 16
3.1.3.6 Strengths and Weaknesses 17
3.1.4 Global State Routing 17
3.1.4.1 Protocol Description 17
3.1.4.2 Complexity comparison with LS and DBF 17
3.1.4.3 Strengths and Weaknesses 18
3.1.5 Fisheye State Routing Algorithm 18
3.1.5.1 Protocol Description 18
3.1.5.2 Complexity 19
3.1.5.3 Strengths and Weaknesses 20
3.1.6 Hierarchical State Routing 20
3.1.6.1 Cluster Formation and Election of Cluster heads 20
3.1.6.2 Physical Multilevel Clustering 20
3.1.6.2.1 Strengths and Weaknesses 21
3.1.6.3 Logical Partitions and Location Management 22
3.1.6.3.1 Strengths and Weaknesses 22
3.2 Reactive (On-demand) Protocols 22
3.2.1 Ad-Hoc On Demand Distance Vector Protocol (AODV) 22
3.2.1.1 AODV Properties 22
3.2.1.2 Unicast Route Establishment 23
3.2.1.2.1 Route Discovery 23
3.2.1.2.2 Expending Ring Search 24
3.2.1.2.3 Forward Path Setup 24
3.2.1.2.4 Route Table Management 24
3.2.1.2.5 Route Maintenance 25
3.2.1.2.6 Local Connectivity Management 25
3.2.1.2.7 Actions after Reboot 26
3.2.1.3 Broadcast 26
3.2.1.4 Security Considerations 26
3.2.1.5 Optimizations And Enhancements 26
3.2.1.4.1 Quality of Service 26
3.2.1.4.2 Subnet Routing 27
3.2.1.5 Complexities 27
3.2.1.6 Strengths and Weaknesses 27
3.2.2 Dynamic Source Routing (DRS) 27
3.2.2.1 Protocol Description 27
3.2.2.2 Route Discovery 28
3.2.2.3 Route Maintenance 28
3.2.2.4 Optimizations 28
3.2.2.5 Integration with the Internet 30
3.2.2.6 Integration with Mobile IP 31
3.2.2.7 Multicast Routing with DSR 32
3.2.2.8 Complexities 32
3.2.2.9 Strengths and Weaknesses 32
3.2.3 Temporally-Ordered Routing Algorithm 33
3.2.3.1 Protocol Description 33

3.2.3.1.1 Creating Routes 33
3.2.3.1.2 Maintaining Routes 35
3.2.3.1.3 Erasing Routes 36
3.2.3.1.4 Optimizing Routes 37
3.2.3.2 Complexities 38
3.2.3.3 Strengths and Weaknesses 38
3.2.4 Associativity-Based Routing (ABR) 38
3.2.4.1 Protocol Description 39
3.2.4.1.1 Route Discovery phase 39
3.2.4.1.2 Route Reconstruction Phase (RRC) 40
3.2.4.1.3 Route Deletion Phase 41
3.2.4.2 Acknowledgement and Retransmission 41
3.2.4.3 Strengths and Weakness 41
3.2.4.4 ABR Protocol Summary 42
3.2.4.5 Complexity Comparison 42
3.2.5 Signal Stability Routing 43
3.2.5.1 Protocol Overview 43
3.2.5.2 Protocol Modules 43
3.2.5.3 Route Maintenance 44
3.2.5.4 Optimization 44
3.2.5.5 Complexities 44
3.2.5.6 Strengths and Weaknesses 44
3.3 Hybrid Routing Protocol 45
3.3.1 Zone Routing Protocol (ZRP) 45
3.3.1.1 Intrazone Routing Protocol (IARP) 45
3.3.1.2 Interzone Routing Protocol (IERP) 45
3.3.1.3 Optimization 47
3.3.1.4 Strengths and Weaknesses 48
4 Unicast Comparison 49
4.1 Table-Driven protocols 49
4.2 Source-Initiated On-Demand Routing Protocols 50
5 Unicast Performance Comparison 54
5.1 Results Comparison 55
5.1.1 Routing Overhead 55
5.1.2 Packet Delivery Ratio 55
5.2.3 End-to-End Delay 56
5.2.4 Path Optimality 56
5.2.5 Throughput 56
5.2.6 Discussion 56
6 Existing Multicast routing Protocol 57
6.1 Ad hoc Multicast Routing Protocol (AMRoute) 57
6.1.1 Assumptions (or lack thereof) 58
6.1.2 Concepts 58
6.1.2.1 User-Multicast Distribution Trees 58
6.1.2.2 Logical core and non-core members 59
6.1.3 Operation 60
6.1.3.1 Control Message 60
6.1.3.2 Mesh Creation 61

6.1.3.3 Joining and Leaving a Group 61
6.1.3.4 Becoming a Passive Non-core node 61
6.1.3.5 Limitation on link connectivity 61
6.1.3.6 Dynamic Core Migration 61
6.1.4 Tree Creation 62
6.1.4.1 Periodic TREE-CREATE Broadcast 62
6.1.4.2 Purging of Tree Links 62
6.1.4.3 Transient Loops 62
6.1.4.4 Effect of Node Failures and Group Leaves 62
6.1.4.5 Core Resolution 63
6.1.4.6 Picking which branch to use for the Tree 63
6.1.4.7 State Diagram 63
6.1.4.8 Timers 64
6.1.4.9 Data Structures 64
6.1.5 Strengths and Weaknesses 64

6.2 Multicast Ad Hoc On-Demand Distance Vector Protocol (MAODV) 65
6.2.1 Route Tables 65
6.2.2 Route Discovery 65
6.2.3 Forward Path Setup 66
6.2.4 Multicast Route Activation 66
6.2.5 Multicast Route Deactivation 66
6.2.6 Multicast Tree Maintenance 67
6.2.6.1 Link Breaks 67
6.2.6.2 Reconnecting Partitioned Trees 68
6.2.7 Actions after Reboot 68
6.2.8 Strengths and Weaknesses 69

6.3 The Core-Assisted Mesh Protocol 69
6.3.1 Overview of CAMP 69
6.3.2 Information used in CAMP 70
6.3.3 Types of cores 71
6.3.4 Joining and Quitting the Multicast Mesh 71
6.3.5 Handling Mobility and Mesh Partitioning 73
6.3.5.1 Link Failures 73
6.3.5.2 Unreachable cores 73
6.3.5.3 Keeping Meshes Connected 73
6.3.6 Strengths and Weaknesses 74

6.4 On - Demand Multicast Routing Protocol (ODMRP) 74
6.4.1 Multicast Route and Mesh Creation 75
6.4.2 Example 76
6.4.3 Reliability 76
6.4.4 Data Forwarding 77
6.4.5 Soft State 77
6.4.6 Selection of Timer Values 77
6.4.7 Unicast Capability 78
6.4.8 Data Structures 78
6.4.8.1 Routing Table 78
6.4.8.2 Forwarding Group Table 78
6.4.8.3 Message Cache 78
6.4.9 Mobility Prediction 78

6.4.9.1 Adapting the Refresh Interval via Mobility Prediction 78
6.4.9.2 Route Selection Criteria 79
6.4.9.3 Alternative Method of Prediction 80
6.4.10 Strengths and Weakness 80

7 Multicast Comparison 81

8 Multicast Performance Comparison 83

9 Challenges of ad hoc wireless networks 85
9.1 Problems remain to be solved at different layers of the protocol stack 85
9.2 Several other issues of interest span various layers of the protocol stack 85
9.2.1 Some of the unanswered questions will be briefly considered here 86

10 Conclusion 92

References 93

Appendix A - Abbreviations and Terminology 97
A.1 Abbreviations 97
A.2 Terminology Used 98

List of Figures

Figure 1. Handoff Between Access Points 2
Figure 2. Mobile IP 2
Figure 3. Example of Ad Hoc Network 2
Figure 4. Routing overhead and mobility for different routing mechanism 6
Figure 5. Categorization of Unicast Ad-Hoc Routing Protocols 7
Figure 6. Example of an Ad Hoc Network 9
Figure 7. Movement in an Ad Hoc Network 10
Figure 8. Example of CGSR routing from node 1 to node 12 14
Figure 9. Example of algorithm's operation 16
Figure 10. Scope of fisheye 18
Figure 11. Message reduction using fisheye 19
Figure 12. An example of physical/virtual clustering 21
Figure 13. Propagation of RREQ throughout the Network 23
Figure 14. Route Determination from Source to Destination 24
Figure 15. Route Maintenance 25
Figure 16. Basic operation of the DSR 28
Figure 17. An example ad hoc network illustrating use of the route cache 29
Figure 18. Mobile host D notices that the route can be shortened 30
Figure 19. RREQ for a node not in the ad hoc network being answered by the FA 31
Figure 20. The roaming node registering with a FA located on G1 in the ad hoc network 31
Figure 21. Creating Routes for TORA 34
Figure 22. Decision Tree for Maintaining Tree 35
Figure 23. A Link Failure Requiring No Reaction 35
Figure 24. Maintaining Routes after a Link Failure Requiring a Reaction 36
Figure 25. Erasing Process Following a Network Partition 37
Figure 26. Time and spatial Representation of Associativity of a MH with its nighbors 38
Figure 27: Updating Assiciativity Metric During BQ Packet 39
Figure 28. REPLY Interruption Caused By Unexpected INs' 40
Figure 29. Route maintenance when SRC, DEST and IN moves 40
Figure 30. Handling Packet Retransmission in ABR 41
Figure 31. Summary of Route Reconstructions under varying node's migration 42
Figure 32. An Example of SSR network 43
Figure 33. Failure of route erase packet 44
Figure 34. Zone of node F of radius 2 45
Figure 35. Network using ZRP 46
Figure 36. Advanced Query Detection (QD1/QD2) 47
Figure 37. Early termination 47
Figure 38. ZRP Zone Routing Radius Optimization 48
Figure 39. Categorization of Ad-Hoc Multicast Routing Protocols 57
Figure 40. A Virtual User-Multicast 59
Figure 41. Simplified AMRoute Node State Diagram 63
Figure 42. Multicast Join Operation 65
Figure 43. Leaving the Multicast Group 66
Figure 44. Repair of a Broken Tree Link 67

Figure 45. Merging of Partitioned Multicast Tree 68
Figure 46. Traffic flow in a multicast mesh and in the multicast shared 69
Figure 47. On-demand procedure for membership setup and maintenance 75
Figure 48. The forwarding group concept 75
Figure 49. Why a mesh 75
Figure 50. An example of a JOIN REPLY forwarding 75
Figure 51. Passive acknowledgements 77
Figure 52. Route selection example 80

List of Tables

Table 1. MH_4 Forwarding Table 10
Table 2. MH_4 Advertised Route Table 10
Table 3. MH_4 Forwarding Table (Updated) 11
Table 4. MH_4 Advertised Route (Updated) 10
Table 5. Comparison of various routing methods 12
Table 6. Example of Routing Table 14
Table 7. Example of Cluster Member Table 14
Table 8. Complexity comparison 19
Table 9. Complexity Comparison of ABR with Some Existing Routing Protocols 27
Table 10. Complexity Comparison of ABR with Some Existing Routing Protocols 42
Table 11. Comparisons of the characteristics of table-driven routing protocols 50
Table 12. Comparisons of the characteristics of on-demand routing protocols 51
Table 13. Comparisons of the characteristics of multicast routing protocols 81

1 Introduction

1.1 Motivation

Since their emergence in the 1970's, wireless communication between mobile users is becoming more popular than ever before. This due to recent technological advances in laptop computers and wireless data communication devices, such as wireless modems and wireless LANs.

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

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

- On demand setup
- Fault tolerance
- Unconstrained connectivity

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

Because nodes are forwarding packets for each other, some sort of routing protocol is necessary to make the routing decisions. Currently there does not any exist any standard for a routing protocol for ad hoc networks, instead this is work in progress. Many problems remain to be solved before any standard can be determined.

1.2 Infrastructured Wireless Networks

The previous section illustrates two distinct types of wireless networks: infrastructured and infrastructureless. *Infrastructured wireless networks*, also known as *cellular networks* have a wired backbone of stationary nodes that are connected to the rest of the network or the Internet. These stationary nodes are often called either base station (BS) or access points (AP). The transmission range of an AP constitutes a cell. All the mobile nodes lying within this cell connects to and communicates with the nearest AP. A *handoff* occurs as a mobile host travels out of range of one AP into that of another and thus, the mobile host is able to continue communication seamlessly throughout the network. Figure 1 illustrates the handoff of a mobile node from one AP to another. Infrastructured wireless networks are commonly used in office buildings and college campuses, or locations where the access points can be easily installed and connected to an existing network.

When a node leaves its local network, routing difficulties commence because the subnet IP address of the node and the network to which it has moved are likely to differ. Hence, the node can no longer receive packets addressed to it. To solve this problem, Mobile IP [4] has been developed. Mobile IP works by distinguishing certain nodes as home agents and others as foreign agents. A home agent is a router on a node's home network that maintains a location registry of mobile nodes while they are connected to other

networks. Foreign agents periodically emits beacons indicating their foreign agent status. A mobile node in a foreign network hears this beacons and registers with the foreign agents, as indicated by the line marked "1" in Figure 2. The foreign agent then contacts the node's home agent (line"2" in the Figure), and a *tunnel* is established whereby packets destined for the mobile node that arrive at the node's home network are encapsulated and then tunneled to the foreign agent on the foreign network (line "3"). Upon receiving the packet, the foreign agent takes the extra header off of the packet and then forwards it to the mobile node (line "4"). Packets initiated by the mobile node are sent directly to the intended recipient.

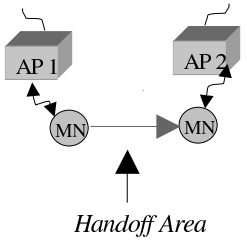


Figure 1. Handoff Between Access Points

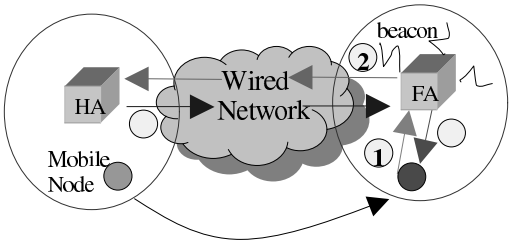


Figure 2. Mobile IP

1.3 Infrastructureless Wireless Networks

In contrast to infrastructured networks, infrastructureless networks, known as mobile ad hoc networks, are a collection of mobile/semi-mobile nodes with no fixed infrastructure and all nodes are capable of movement. Communication in an ad hoc network is peer-to-peer as the mobile nodes communicate directly with one another. Because the transmission range of the nodes is likely to be limited, it is often the case that multiple hops are needed to connect a given source and a destination. Hence, the nodes must serve as routers for other nodes in the network so that data packets can be forwarded to their destinations. Figure 3 illustrates an example of an ad hoc network.

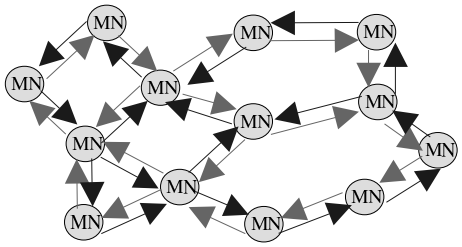


Figure 3. Example of Ad Hoc Network

Ad hoc mobile networks generally have reduced administrative cost compared to wired networks. These networks are self-configuring, and so they are able to maintain network connections and routing information without the need for explicit route setup by a system administrator.

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

1.4 Applications

With the increase of portable devices as well as progress in wireless communication, the number of applications for ad hoc network has increased. Ad hoc networking can be applied anywhere where there are wireless devices connected to a network.

In areas where no infrastructure such as the Internet is available an ad-hoc network could be used by a group of wireless mobile hosts. This can be the case in areas where a network infrastructure may be

undesirable due to reasons such as cost or convenience. Examples of such situations include disaster recovery personnel or military troops in cases where the normal infrastructure is either unavailable or destroyed.

Other examples include business associates wishing to share files in an airport terminal, or a class of students needing to interact during a lecture and conference scenarios where attendees want to be able to easily share files. If each mobile host wishing to communicate is equipped with a wireless local area network interface, the group of mobile hosts may form an ad-hoc network. Ad hoc networks are also suitable for networking in locations without an existing wired infrastructure, such as data collection in open fields, and sensor networks. Access to the Internet and access to resources in networks such as printers are features that probably also will be supported.

1.5 Characteristics

Mobile nodes have many unique characteristics that make traditional routing protocols inapplicable. Ad-hoc networks are often characterized by a dynamic topology due to the fact that nodes change their physical location by moving around. This favors routing protocols that dynamically discover routes over conventional routing algorithms like distance vector and link state [39]. Another characteristic is that a host/node has very limited CPU capacity, storage capacity, battery power and bandwidth, also referred to as a "thin client". This means that the power usage limits the radio transmission range.

The access media, the radio environment, also has special characteristics that must be considered when designing protocols for ad-hoc networks. One example of this may be unidirectional links. These links arise when for example two nodes have different strength on their transmitters, allowing only one of the hosts to hear the other, but can also arise from disturbances from the surroundings. Multihop in a radio environment may result in an overall transmit capacity gain and power gain, due to the squared relation between coverage and required output power. By using multihop, nodes can transmit the packets with much lower output power. The protocol should also be self-starting, and it should be loop free at all times, because even a temporary routing loop wastes the scares bandwidth.

1.6 Desirable properties

If the conventional routing protocols do not meet our demands, we need a new routing protocol. The question is what properties should such protocols have? These are some of the properties [45] that are desirable:

Distributed operation: The routing protocol must be distributed in order to increase reliability. Where all nodes are mobile, it is unacceptable to have a centralized routing protocol. Each node should be intelligent enough to make routing decisions using other collaborative nodes. A distributed but virtually centralized protocol is a good idea.

Loop free: To improve the overall performance, we want the routing protocol to guarantee that the routes supplied are loop-free. This avoids any waste of bandwidth or CPU consumption.

Demand based operation: The routing algorithm should adapt to traffic on demand or need basis for efficient utilization of network energy and bandwidth resources. This means that the protocol should only react when needed and that the protocol should not periodically broadcast control information. Though the obvious drawback is increased delay.

Proactive operation: In some contexts, additional latency incurred due to demand based operation, may be unacceptable. So, if the resources bandwidth permits, proactive operation must be used.

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

Security: The radio environment is especially vulnerable to impersonation attacks, so to ensure the wanted behavior from the routing protocol, we need some sort of preventive security measures. Authentication and encryption is probably the way to go and the problem here lies in distributing keys among the nodes in the ad-hoc network. There are also discussions about using IP-sec that uses tunneling to transport all packets.

Power conservation: The nodes in an ad-hoc network can be laptops and thin clients, such as PDAs that are very limited in battery power and therefore uses some sort of stand-by mode to save power. It is therefore important that the routing protocol has support for these sleep-modes.

Multiple routes: Multiple routes could be used to reduce the number of reactions to topological changes and congestion. If one route has become invalid, it is possible that another stored route could still be valid and thus saving the routing protocol from initiating another route discovery procedure.

Quality of service support: Some sort of quality of service support is probably necessary to incorporate into the routing protocol. This is related with what these networks will be used for. It could for instance be real-time traffic support.

None of the proposed protocols from MANET have all these properties, but it is necessary to remember that the protocols are still under development and will probably be extended with more functionality. The remainder of this report will describe the different routing protocols and analyze them theoretically

1.7 Description

This report consists of 10 chapters and one appendix. Chapters 1 and 2 explain the concept of ad hoc networks and routing in general. Chapters 3 to 8 describe the different unicast and multicast routing protocols designed for these ad hoc networks by first describing the operation of each of the protocols then comparing their various characteristics, relative strengths and weakness, and analyze performance for each of them. Some protocols (specially SSR and CAMP) are not well documented. They could not explained in the same depth as the other protocols described in this report. Challenges facing ad hoc mobile wireless networks are discussed in chapter 9. Chapter 10 concludes the whole report and the lists references that I have used. The appendix contains some terminology and abbreviations used in this report.

2 Routing

A routing protocol is needed to make the routing decision for routing protocols are needed even for fixed networks. The routing protocol has two main functions, selection of routes for various source-destination pairs and the delivery of messages to their correct destination. The second function is conceptually straightforward using a variety of protocols and data structures (routing tables).

2.1 Conventional protocols

The conventional routing protocols like link state or distance vector are well tested and familiar to most computer communications people. If a routing protocol is needed, why not use them? The main problem with them is that they would have problems to converge to a steady state in an ad hoc network with a very frequently changing topology as they are designed for a static topology. Link state and distance vector would probably work very well in an ad-hoc network with low mobility. The problem that still remains is that they are highly dependent on periodic control messages, which require frequent updates of routing tables among the network nodes and thus are costly in resources such as bandwidth, battery power and CPU. Since both link-state and distance vector tries to maintain routes to all reachable destinations, it is necessary to maintain these routes and this also wastes resources for the same reason as above.

Another characteristic of conventional protocols is that they assume bi-directional links. In the wireless radio environment this is not always the case. Because many of the proposed ad-hoc routing protocols have a traditional routing protocol as underlying algorithm, it is necessary to understand the basic operation for conventional protocols like distance vector, link state and source routing.

2.2 Link State

In-link state routing [39], each node maintains a view of the complete topology with a cost for each link. To keep this cost consistent, each node periodically broadcasts the link costs of its outgoing links to all other nodes using flooding. As each node receives this information, it updates its view of the network topology and applies a shortest path algorithm to choose the next-hop for each destination. Some link costs at any particular node can be incorrect because of long propagation delays, partitioned networks, etc. Such inconsistent network topology views can lead to the formation of routing-loops. These loops are however short-lived, because they disappear in the time it takes a message to traverse the diameter of the network.

2.3 Distance Vector

In distance vector [39] each node only monitors the cost of its outgoing links, but instead of broadcasting this information to all nodes, it periodically broadcasts to each of its neighbors an estimate of the shortest distance to every other node in the network. The receiving nodes then use this information to recalculate the routing table, by simple comparison of own distance vector to received information. Compared to link-state, distance vector is computationally more efficient, easier to implement and requires much less storage space. However, it is well known that distance vector can cause the formation of both short-lived and long-lived routing loops. The primary cause for this is that the nodes choose their next-hops in a completely distributed manner based on information that can be stale.

2.4 Source Routing

Source routing [39] means that each packet must carry the complete path that the packet should take through the network. The routing decision is therefore made at the source. The advantage with this approach is that it is very easy to avoid routing loops. The disadvantage is that each packet requires a slight overhead.

2.5 Flooding

Many routing protocols use broadcast to distribute control information, that is, to send the control information from an origin node to all other nodes. A widely used form of broadcasting is flooding [39] and operates as follows. The origin node sends its information to its neighbors (in the wireless case, this means all nodes that are within transmitter range). The neighbors relay it to their neighbors and so on, until the packet has reached all nodes in the network. A node will only relay a packet once and to ensure this some sort of sequence number can be used. This sequence number is increased for each new packet a node sends.

2.6 Classification

Routing protocols can be classified [49] into different categories depending on their properties.

- Centralized vs. Distributed
- Static vs. Adaptive
- Reactive vs. Proactive

From this classification, only the last one is really pertinent to our study. It is patently obvious that in our context, any routing mechanism that relies on centralised control is out of the question. The same applies to all static routing schemes, static meaning here fixed routes to destinations without any response to changing traffic conditions

This method of classification classifies the routing algorithms as either proactive or reactive. Proactive protocols attempt to continuously evaluate the routes within the network, so that when a packet needs to be forwarded, the route is already known and can be immediately used. In proactive algorithm, there is always constant number of transmissions going, even when network is in equilibrium, i.e. routing overhead is constant with regard to non-data routing packets. The family of Distance-Vector protocols is an example of a proactive scheme. Reactive protocols, on the other hand, invoke a route determination procedure on demand only. Thus, when a route is needed, some sort of global search procedure is employed. The family of classical flooding algorithms belongs to the reactive group. Figure 4 illustrates routing overhead vs. mobility for different routing mechanisms.

The advantage of the proactive schemes is that, once a route is needed, there is little delay until the route is determined. In reactive protocols, because route information may not be available at the time a route request is received, the delay to determine a route can be quite significant. Furthermore, the global search procedure of the reactive protocols requires significant control traffic, pure reactive routing protocols may not be applicable to real-time application.

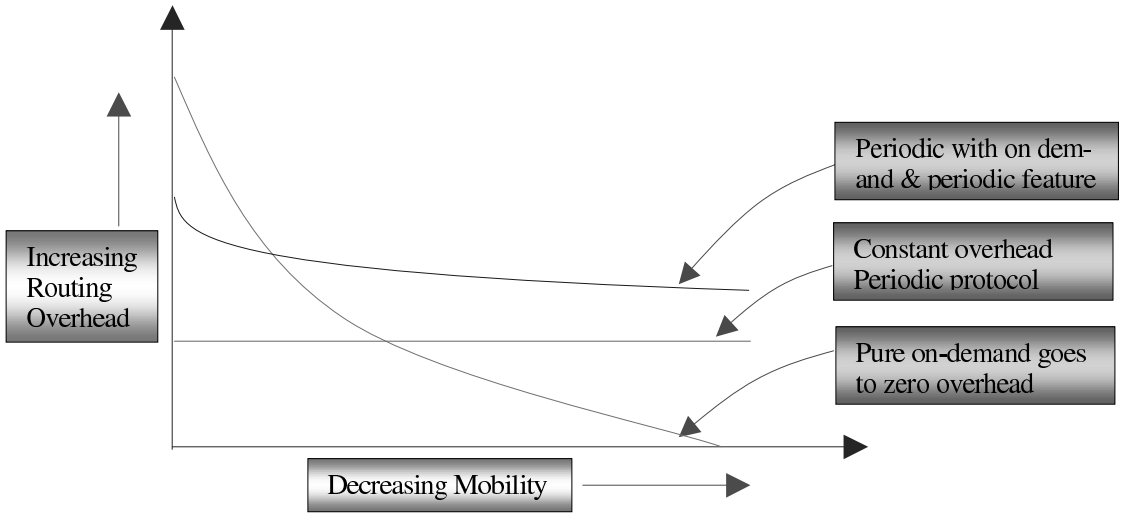


Figure 4. Routing overhead and mobility for different routing mechanism

3 Existing Unicast Routing Protocols

There are three categories of unicast routing protocols. Proactive, or table-driven, protocols attempt to maintain consistent, up-to-date information for all destinations on each node. Examples of table-driven protocols are CGSR, WRP, DSDV, CGSR, FSR etc. Reactive or on-demand protocols attempt to minimize overhead by discovering routes on demand. Examples are TORA, ABR, DSR, AODV, CBRO etc. Lastly, hybrid protocols combine aspects of proactive and reactive protocols. Examples are ZRP, ZHLS.

Unicast routing schemes can be subdivided into two further categories. Flat and Hierarchical. In flat routing schemes, each node maintains a routing table with entries for all the nodes in the network. In this category, many protocols have been proposed to support mobile ad hoc wireless routing. Some proposals are extensions of schemes previously developed for traditional wired networks. For example, Perkin's destination sequence distance vector (DSDV) [5] is based on distributed Bellman-Ford (DBF), Garcia's wireless routing protocol (WRP) [50,57] is based on a loop-free path-finding algorithm, etc. Flat routing scheme is acceptable if the user population is small. However, as the number of mobile hosts increases, so does the overhead. Thus, flat routing algorithms do not scale well to large network.

To permit scaling, hierarchical techniques can be used. In these techniques, the network contains two kinds of nodes, endpoints and switches. Only endpoints can be sources and destinations for user data traffic, and only switches can perform routing functions. To form the lowest level partitions in the hierarchy, endpoints choose the most convenient switches to which they will associate by checking radio link quality. Autonomously, they group themselves into cells around those switches (cluster heads). This procedure is called "cell formation". Each endpoint is within one hop of the switch with which it is affiliated. The switches, in turn, organize themselves hierarchically into clusters, each of which functions as a multihop packet-radio network. First-level cluster heads organize to form higher-level clusters and so on. This procedure is called "hierarchical clustering." (A switch is a zeroth level cluster.) As nodes moves, clusters may split or merge, altering membership.

The major advantage of hierarchical routing is the efficient utilization of radio channel resources and the drastic reduction of routing table storage, transmission and processing overhead. A hierarchical clustering and routing approach specifically designed for large wireless networks and addresses the link and network layers only and is independent of the physical/MAC layer.

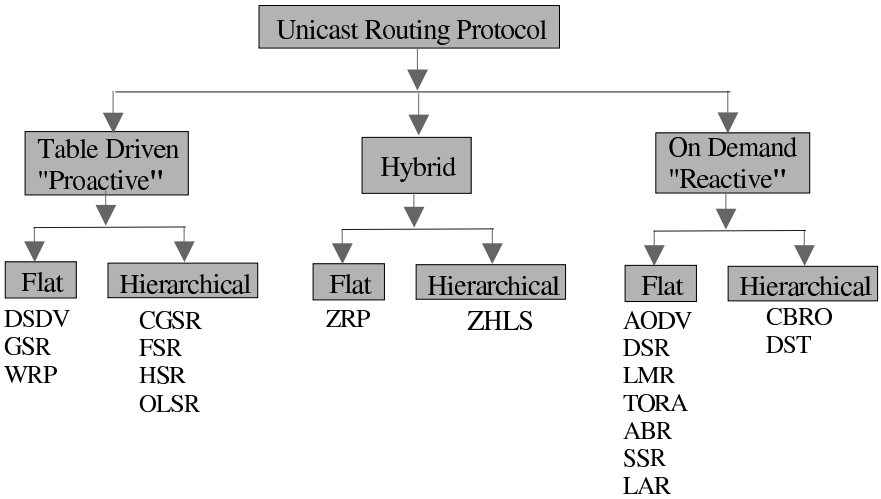


Figure 5. Categorization of Unicast Ad-Hoc Routing Protocols

3.1 Proactive (Table - driven) Protocols

In Table-driven routing protocols each node maintains one or more tables containing routing information to every other node in the network. All nodes update these tables so as to maintain a consistent and up-to-date view of the network. When the network topology changes the nodes propagate update messages throughout the network in order to maintain consistent and up-to-date routing information about the whole network. These routing protocols differ in the method by which the topology change information is distributed across the network and the number of necessary routing-related tables. The following sections discuss some of the existing table-driven ad hoc routing protocols.

3.1.1 Destination Sequenced Distance Vector (DSDV)

Destination-Sequenced Distance-Vector Routing (DSDV) [5] is an adaption of a conventional routing protocol to ad hoc networks. It is derived from a classical distance vector algorithm, Distributed Bellman-Ford (DBF) algorithm [13]. Enhancements are made in order to avoid the looping problem present in the basic DBF. Formation of loops is avoided by tagging each route table entry with a sequence number to order the routing information.

3.1.1.1 Protocol Overview

In DSDV, packets are routed between nodes of an ad hoc network using routing tables stored at each node. Each routing table, at each node, lists all available destinations and the number of hops to each. Each route table entry is tagged with a sequence number that is originated by the destination. To maintain the consistency of the routing tables, DSDV uses both periodic and triggered routing updates; triggered routing updates are used in addition to the periodic updates in order to propagate the routing information as quickly as possible when there is any topological change. The update packets include the destinations accessible from each node and the number of hops required to reach each destination along with the sequence number associated with each route. Data is also kept about the length of time between the arrival of the first and the arrival of the best route for each particular destination. On the basis of this data, a decision may be made to delay advertising routes that are about to change, thus damping fluctuations of the route tables. The advertisement of possibly unstable routes is delayed to reduce the number of rebroadcasts of possible route entries that normally arrive with the same sequence number.

3.1.1.2 Route Advertisements

The DSDV protocol requires each mobile node to advertise, its own route table to each of its current neighbors. The advertisement must be made often enough to ensure that every mobile node can almost always locate every other mobile node in this collection as the entries in the list may change fairly dynamically over time. In addition, each mobile node agrees to relay data packets to other nodes upon request. This agreement places a premium on the ability to determine the shortest number of hops for a route to a destination. In this way a mobile node may exchange data with any other mobile node in the group even if the target of the data is not within range for direct communication. All the nodes interoperating to create data paths between themselves broadcast the necessary data periodically, on the order of every few seconds.

3.1.1.3 Route Table Entry and Route Selection Criteria

The data broadcast by each node will contain its new sequence number and the following information for each new route:

- The destination's address

- The number of hops required to reach the destination
- The sequence number of the information received regarding that destination, as originally stamped by the destination.

Whenever a node X comes up, it broadcasts a beacon message ("I am alive message") stamping it with a locally maintained sequence number. The nodes in its neighborhood listen to this message and update the information for this node. If the nodes do not have any previous entry for this node X, they simply enter X's address in their routing table, together with hop count (in this case it is 1) and sequence number as broadcasted X. If the nodes had a previous entry for X, the sequence number of the broadcasted information is compared to the sequence number stored in the node for destination X. If the message received has a higher sequence number, then this means that the node X has propagated a new information about its location so the entry must be updated in accordance with the new information received. The information with a newer sequence number is definitely new as the node X itself stamps sequence number.

The new information that a node receives is scheduled for broadcasting to its neighbors so that they know about the changes in topology. The neighboring nodes also follow the same rule i.e. updating the information when an information about a node with a newer sequence number is received. The metrics for routes chosen from the newly received broadcast information are each incremented by one hop. So, the new information is updated gradually at all nodes and they now know the next hop node in order to correctly route the packet to destination X.

3.1.1.4 Responding to Topology Changes

Mobile nodes cause broken links as they move from place to place. When a link to the next hop is broken, any route through that next hop is immediately assigned an infinity metric and assigned an updated sequence number. This is the only case when sequence numbers are assigned by somebody other than the destination. When a node receives an infinity metric, and it has an equal or later sequence number with a finite metric, a route update broadcast is triggered. Therefore, routes with infinity metric will be quickly replaced by real routes propagated from the newly located destination. To avoid nodes and their neighbors generating conflicting sequence numbers when the topology changes, nodes only generate even sequence numbers for themselves, and neighbors responding to link changes only generate odd sequence numbers.

In order to reduce the network overhead, a node can choose two packet types: *full dump* that *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 the incremental updates.

3.1.1.5 Extending Base Station Coverage

Mobile nodes will frequently be used in conjunction with base stations, which allow them to exchange data with other nodes connected to the wired network. By participating in the DSDV protocol, base stations can extend their coverage beyond the range imposed by their wireless transmitter. When a base station participates in DSDV, it is shown as a default route in the table transmitted by a mobile node. In this way, mobile nodes within range of a base station can cooporate to effectively extend the base station range to serve other nodes out of that range, as long as those other mobile nodes are close to one of the mobile nodes that are within range.

3.1.1.6 Examples of DSDV in Operation

Table 1 shows a possible structure of the forwarding table maintained at MH_4 for the network topology shown in Figure 6. Suppose the address of each nodes MH_i, all sequence numbers are denoted SNNN_MH_i. The install time field helps determine when to delete a stale route.

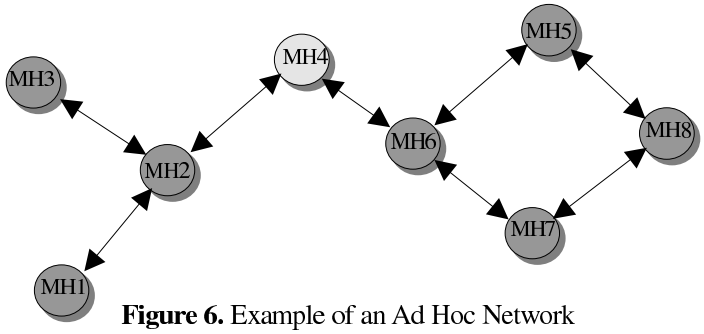


Figure 6. Example of an Ad Hoc Network

Destination	Next Hop	Metric	Seq. Number	Install	Stable
MH_1	MH_2	2	S406_MH_1	T001_MH_4	Ptr1_MH_1
MH_2	MH_2	1	S128_MH_2	T001_MH_4	Ptr1_MH_1
MH_3	MH_2	2	S564_MH_3	T001_MH_4	Ptr1_MH_1
MH_4	MH_4	0	S710_MH_4	T001_MH_4	Ptr1_MH_1
MH_5	MH_6	2	S392_MH_5	T002_MH_4	Ptr1_MH_1
MH_6	MH_6	1	S076_MH_6	T001_MH_4	Ptr1_MH_1
MH_7	MH_6	2	S128_MH_7	T002_MH_4	Ptr1_MH_1
MH_8	MH_6	3	S050_MH_8	T002_MH_4	Ptr1_MH_1

Table 1. MH_4 Forwarding Table

Courtesy from Carleton university lecture note (94.581x)

From Table 1 we can see that all of the nodes become available to MH_4 at about the same time because, for most of them, their Install_Time is about the same.

Table 2 shows the structure of the advertised route table of MH_4.

Destination	Metric	Seq. Number
MH_1	2	S406_MH_1
MH_2	1	S128_MH_2
MH_3	2	S564_MH_3
MH_4	0	S710_MH_4
MH_5	2	S392_MH_5
MH_6	1	S076_MH_6
MH_7	2	S128_MH_7
MH_8	3	S050_MH_8

Table 2. MH_4 Advertised Route Table

Destination	Metric	Seq. Number
MH_4	0	S820_MH_1
MH_1	3	S516_MH_2
MH_2	1	S238_MH_3
MH_3	2	S674_MH_4
MH_5	2	S502_MH_5
MH_6	1	S186_MH_6
MH_7	2	S238_MH_7
MH_8	3	S160_MH_8

Table 4. MH_4 Advertised Route (Updated)

Now suppose that MH_1 moves into the general vicinity of MH_8 and MH_7 and away from the others (especially MH_2).

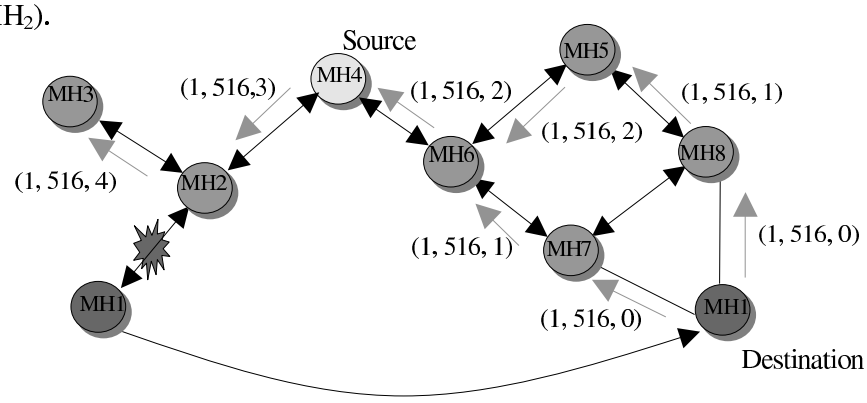


Figure 7. Movement in an Ad Hoc Network.

Above Fig. shows DSDV propagation of route information for destination MH1. Each tuple represents the destination address, sequence number, and link metric (hop count)

Route advertisement begins at the destination node and propagates through the entire network. Figure 7 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 MH_1. In reality, each node would broadcast its entire route table, not just the information for MH_1.The new internal forwarding tables at MH_4 might then appear as shown in Table 3

Table 3 shows only the entry for MH_1 shows a new metric, but in the intervening time many new sequence number entries have been received. This change happens solely because of the higher sequence number. The same update message (i.e. same destination and sequence number) arrives at MH_6 from MH_5 and MH_7 respectively. MH_6 chooses the message from MH_7 because it has a better metric.

Destination	Next Hop	Metric	Seq. Number	Install	Stable Data
MH_1	MH_6	3	S516_MH_1	T810_MH_4	Ptr1_MH_1
MH_2	MH_2	1	S238_MH_2	T001_MH_4	Ptr1_MH_1
MH_3	MH_2	2	S674_MH_3	T001_MH_4	Ptr1_MH_1
MH_4	MH_4	0	S820_MH_4	T001_MH_4	Ptr1_MH_1
MH_5	MH_6	2	S502_MH_5	T002_MH_4	Ptr1_MH_1
MH_6	MH_6	1	S186_MH_6	T001_MH_4	Ptr1_MH_1
MH_7	MH_6	2	S238_MH_7	T002_MH_4	Ptr1_MH_1
MH_8	MH_6	3	S160_MH_8	T002_MH_4	Ptr1_MH_1

Table 3. MH_4 Forwarding Table (Updated)

At MH_4, the incremental advertised routing update has the form shown in Table 4 In this advertisement, the information for MH_4 comes first, since it is doing the advertisement. The information for MH_1 comes next as MH_1 is the only one that has any significant route changes affecting it.

3.1.1.6.1 Damping Fluctuations

DSDV also employs a mechanism to damp in fluctuations in route table updates. In an environment where many independent nodes transmit routing information asynchronously, some fluctuation could develop. For example, a node could receive two routes to the same destination with the same sequence number, however, the one with the worse metric always arrives first. This could lead to continues outbursts of route updates and fluctuations of the routing tables. DSDV solves this problem by using the idea of "settling time". Specifically, the time duration until the route becomes stable (termed settling time) is predicted, and the settling time is allowed before advertising any new route information to the network. In other words, the settling time is used to decide how long to wait before advertising new routes. By delaying the advertisement of unstable routes, fluctuations of the routing tables are prevented, and consequently, the number of route updates are reduced.

3.1.1.7 Properties of the DSDV Protocol

DSDV claims the following properties:

- Loop-free at all instants
- Dynamic, multi-hop, self-starting
- Low memory requirements
- Quick convergence via triggered updates

- Routes available for all destinations
- Fast processing time
- Reasonable network load
- Minimal route trashing
- Intended for operation with up to 100 mobile nodes, depending on "mobility factor".

3.1.1.7.1 Proof of loop-free property

Potentially a loop may form each time node i changes it's next hop. This can happen in two cases:

- Node i detects that the link to its next-hop is broken: Clearly, this action cannot form a loop involving i.
- Node i receives from one of its neighbors k, a route to D, with a sequence number s(k) such that s(k) > s(i) the sequence number for the Destination D as originally stored in node i.

A node i propagates sequence number s(i) to its neighbors only after receiving it from its current next hop. When a node i chooses a particular neighbor as the next hop for a particular destination, the information for this is received by that next hop only. So, at all times the sequence number value stored at the next hop is always greater or equal to the value stored at i. Starting from node i, if we follow the chain of next-hop pointers, the sequence number values stored at visited nodes would form a non decreasing sequence. Now suppose, node i forms a loop by choosing k as its next-hop. This would imply node i lies both before and after k. Since it lies after k, thus we must have s(k) <= s(i) (as proved that sequence numbers along a path form a non decreasing sequence). But this contradicts the initial assumption that s(k) > s(i). Hence, loop formation cannot occur if nodes use newer sequence numbers to pick routes.

3.1.1.8 Comparison with Other Methods

Table 5 presents a quick summary of some of the main features of a few chosen routing protocols. The primary concern with using a Distributed Bellman Ford algorithm in ad hoc environment is its susceptibility towards forming routing loops and the counting-to-infinity problem. RIP is also suffers from the same problem. Unlike DBF, RIP only keeps track of the best route to each destination and employs techniques known as *split-horizon* and *poisoned-reverse* to avoid ping-pong style of looping, but these techniques can are not powerful enough to avoid loops involving more than two hops. The uncoordinated modifications problem, which is the cause of loop formation in BF, is alleviated by employing an internodal coordination mechanism as proposed by Merlin and Segall in [47]. A similar technique, but with better convergence results, is developed by Jaffe and Moss in [36]. Link-state [37] algorithms are also free of the counting-to-infinity problem. However, they need up-to-date information about the entire network topology at every node. Link State (LS) does not eliminate the creation of temporary routing-loops. The DSDV approach offers a very attractive combination of desirable features. Its memory requirement is very moderate O(n). It guarantees loop free paths at all instances.

Routing Method	Looping	Internal Coordination	Space Complexity
Bellman Ford	s/l	-	O(nd)
Link State	s	-	$O(n^2)$
Loop-free BF	s	-	O(nd)
RIP	s/l	-	O(n)
Merlin Segall	loop free	Required	O(nd)
Jaffe Moss	loop free	Required	O(nd)
DSDV	loop free	-	O(n)

s - short term loop, l - long term loop
n - number of nodes, d - maximum degree of node

Table 5. Comparison of various routing methods

3.1.1.9 Strengths and Weaknesses

Strengths: One of the major advantages of DSDV is that it provides loop-free routes at all instants and memory requirement is moderate O(n).

Weaknesses: DSDV has a number of drawbacks. Optimal values for the parameters like maximum settling time for a particular destination are difficult to determine. This might lead to route fluctuations and spurious advertisements resulting in waste of bandwidth. DSDV also uses both periodic and triggered routing updates, which could cause excessive communication overhead. In addition, in DSDV, a node has to wait until it receives the next route update originated by the destination before it can update its routing table entry for that destination. This implicit destination-centered synchronization suffers a latency problem [Murt95]. Furthermore, DSDV does not support multipath routing.

3.1.2 Clusterhead Gateway Switch Routing (CGSR) Protocol

Clusterhead Gateway Switch Routing (CGSR) uses as basis the DSDV Routing algorithm. CGSR protocol differs from the DSDV routing protocol in the type of addressing and network organization scheme employed. Instead of a flat network, CGSR is a clustered multihop mobile wireless network with several heuristic routing schemes.

3.1.2.1 Cluster Formation and Election of Cluster heads

The cluster head election algorithms have been proposed so far: *identified-based clustering* [14, 15, 1] and *connectivity-based clustering* [42, 2]. Each admits both a centralized and distributed implementation. With the centralized version, the node with the lowest- or highest-numbered identifier (identified-based clustering) or with the largest number of neighbors (connectivity-based clustering) is chosen as a clusterhead for the cluster containing that node and its one-hop neighbors. All nodes in the cluster thus defined are excluded from further consideration, and the process of clusterhead selection and cluster formation repeats with the remaining nodes in the network until all nodes are contained within some cluster.

With the distributed version of identifier-based clustering, a node elects itself as a clusterhead if it has the lowest- or highest-numbered identifier in its neighborhood. Otherwise, it elects the bidirectionally connected neighbor with the lowest- or highest-numbered identifier unless that node has relinquished clusterhead status to another node. With the distributed version of connectivity-based clustering, a node becomes a clusterhead if it is the most highly connected of all of its uncovered neighbors. Any node that has not yet elected its clusterhead is said to be uncovered. If a node has already elected another node as its clusterhead, it cannot become a clusterhead itself. Clusterhead ties are resolved according to lowest- or highest-numbered node identifiers.

3.1.2.2 Gateways

A gateway node is a node that is in the communication range of two or more cluster heads. Any node with links to more than one cluster is a candidate gateway connecting these clusters. If the node has two clusterheads as neighbors, it is a candidate gateway connecting two overlapping clusters. If the node has one clusterhead as a neighbor and can reach a second clusterhead in two hops, it is a candidate gateway linked to a candidate gateway in another cluster, which together connect two disjoint clusters. Figure 8 illustrates only overlapping cluster. For overlapping clusters, the gateway selected is the one with the highest- or lowest-numbered identifier. For disjoint clusters, the two gateways selected are the linked pair in which one member has the highest- or lowest-numbered identifier among all candidates connecting the two clusters. Unambiguous selection of a single gateway pair joining two clusters may require advertisement of node identifiers beyond one-hop neighbors.

3.1.2.3 Clustering

The mobile nodes are aggregated into clusters. The cluster is controlled by a cluster head which provides a convenient framework for the development of important features such as code separation among clusters, channel access routing and bandwidth allocation. Using a distributed algorithm within a cluster, a node is elected to be the cluster head. All nodes that are in the communication range of the cluster head belong to its cluster i.e. all nodes in a cluster can communicate with a cluster head and possibly with each other. The most important criteria is stability. The disadvantage of having a cluster head scheme is that frequent cluster head changes can adversely affect routing protocol performance since nodes are busy in cluster head selection rather than packet relaying. Hence, instead of invoking cluster head reselection every time the cluster membership changes, a Least Cluster Change (LLC) clustering algorithm is introduced. In LCC, cluster head change occurs only if a change in network causes two cluster heads to come into one cluster or one of the nodes moves out of the range of all the cluster heads. This is an improvement (in stability) over two previous algorithms, lowest-ID algorithm and highest-connectivity (degree), which reelect the clusterhead every time the cluster membership changes. The LCC algorithm uses either lowest-ID or highest-connectivity for initialization and route maintenance.

3.1.2.4 Routing

CGSR uses DSDV as the underlying routing scheme, and hence has much of the same overhead as DSDV. However, it modifies DSDV by using a hierarchical cluster head-to-gateway routing approach to route traffic from source to destination. The general algorithm works in the following manner. The source

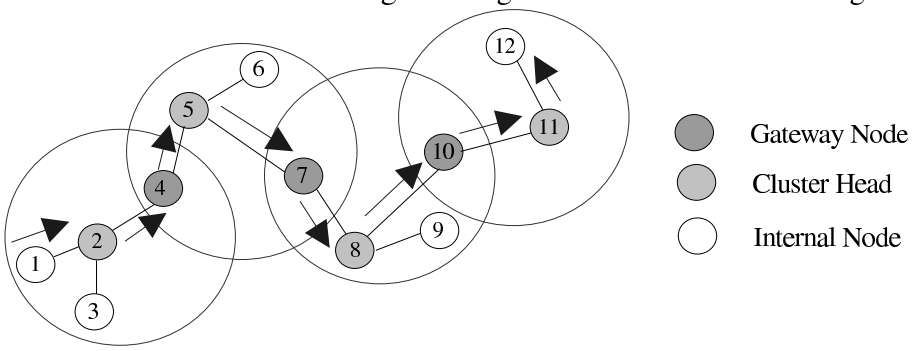


Figure 8. Example of CGSR routing from node 1 to node 12

of the packet transmits the packet to its cluster head. From this cluster head, the packet is sent to the gateway node that connects this cluster head and the next cluster head along the route to the destination. The gateway sends it to that cluster head and so on till the destination cluster head is reached in this way. The destination cluster head then transmits the packet to the destination, which is within its cluster. Figure 8 shows an example of CGSR routing scheme for overlapping cluster.

Destination Cluster	Next Hop	Metrics	Sequence#
11	2	7	71

Table 6. Example of Routing Table

Destination	Destination Clustered	Sequence#
12	11	100

Table 7. Example of Cluster Member Table

Each node maintains a cluster member table (see Table 7) that has a mapping from each node to its respective cluster head. Each node broadcasts its cluster member table periodically and updates its table after receiving other nodes broadcasts using the DSDV algorithm. In addition, each node also maintains a routing table (see Table 6) that determines the next hop to reach the destination cluster. Both tables contain sequence numbers to purge stale routes and prevent looping.

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

3.1.2.5 Strengths and Weaknesses

Strengths: The major advantage of CGSR is that only the routes to the cluster heads are maintained due to the hierarchical routing.

Weaknesses: Some nodes, such as cluster heads and gateway nodes have higher computation and communication burden than other nodes. The network reliability may also be affected due to single points of failure of these critical nodes. Another disadvantage of CGSR is overhead associated with maintaining clusters. Specifically, each node needs to periodically broadcast its cluster member table and update its table based on the received updates.

3.1.3 The Wireless Routing Protocols (WRP)

3.1.3.1 Protocol Overview

Wireless routing protocol [50] is a path finding algorithm is also based on distance vector algorithm. To avoid the looping problem present in the distance vector algorithm, WRP includes the length and second-to-last hop (predecessor) information of the shortest path to each destination

To describe WRP, the network is modeled as an undirected graph G (V, E), where V is the set of nodes and E is the set of links (or edges) connecting the nodes. An undirected edge connecting two nodes is formed if there is radio connectivity between them and they can exchange update messages reliably with a certain probability of success. When a link fails, the corresponding distance entries in a node's distance and routing table are marked as infinity.

Due to changes in radio connectivity or jamming, the messages may be lost or corrupted. After receiving an update message free of errors, a node is required to send a positive acknowledgement (ACK) indicating that it has good radio connectivity and has processed the update message. A node can send a single update message to inform all its neighbors about changes in its routing table because of the broadcast nature of the radio channel,; however, each such neighbor sends an ACK to the originator node.

A periodic update message are sent without any changes of routing table if no recent transmissions of routing table updates or ACKs are received, to ensure the connectivity with the neighbor The time interval between two such null update messages is the *HelloInterval*. If a node fails to receive any type of message from a neighbor for a specified amount of time (e.g., three or four times the HelloInterval known as the *RouterDeadInterval*), the node must assume that connectivity with that neighbor bas been lost.

3.1.3.2 Information Maintained at Each Node

For the purpose of routing, each node in the network is responsible for maintaining four tables:

- Distance table
- Routing table
- Link-cost table
- Message retransmission list (MRL) table

Distance table of a node i contains the distance of each destination node j via each neighbor k of i. It also contains the downstream neighbor of k through which this path is realized. Routing table of node i contains the distance of each destination node j from node i, the predecessor and the successor of node i on this path. It also contains a tag to identify if the entry is a simple path, a loop or invalid. By storing predecessor and successor into the table helps in detecting loops and thus counting-to-infinity problem. Link cost table contains cost of link to each neighbor of the node and the number of timeouts since last

error-free message was received from that neighbor. Each entry of the MRL contains the sequence number of the update message, a retransmission counter, an acknowledgement-required flag vector with one entry per neighbor, and a list of updates sent in the update message. The MRL records which updates in an update message need to be retransmitted and which neighbors should acknowledge the retransmission.

3.1.3.3 Information Exchanged among Node

On link changes and at periodic times nodes exchange routing tables with their neighbors and contains a list of updates (the destination, a distance to the destination and a predecessor to the destination) as well as a list of responses indicating which mobiles should acknowledge (ACK) the updates. An update message also contains the identifier of the sending node and a sequence number assigned by the sending node. Upon receiving an update packet, each node updates the distance and the predecessor entries in the distance table.

3.1.3.4 Routing Table Updating

A node can decide to update its routing table after either receiving an update message from a neighbor, or detecting a change in the status of a link to a neighbor.

This algorithm checks for consistency of all its neighbors every time it detects a change in link of any of its neighbors. Because of this unique feature of WRP, temporary looping is eliminated and convergence is fast.

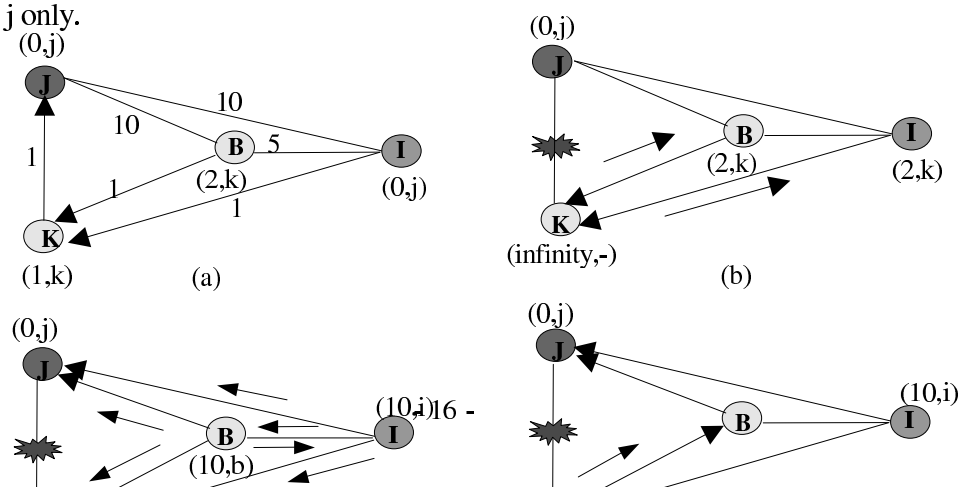
3.1.3.4.1 Handling Topology and Link-Cost Changes

Nodes learn of the existence of their neighbors from the receipt of acknowledgements and other messages. If a node is not sending messages, it must send a *hello message* within a specified time period to ensure connectivity. Otherwise, the lack of messages from the node indicates the failure of that link; this may cause a false alarm. When a node receives a *hello* message from a new node, that new node is added to it's routing table, and the node sends the new node a copy of its routing table information.

In the event of the loss of a link between two nodes, the nodes send update messages to their neighbors. On receiving an update message, node modifies its distance table and looks for better paths using this new information. Any newly found path is sent back to original nodes so that they can update their tables. Node also updates its routing table if new path is better than existing one.

3.1.3.5 Examples

The following example illustrates the working of WRP. Consider a four node network shown in Figure 9(a). All links and nodes are assumed to have the same propagation delays. Link-costs are as indicated in the figure. Node i is the source node, j is the destination node and nodes k and b are the neighbors of node i. The arrows next to links indicate the direction of updates messages and the label in parentheses gives the distance and the predecessor to destination j. Each update will be acknowledged by an ACK message from the neighbor. ACKs are not shown in the figure. The figure focuses on update messages to destination j only.



When link (j, k) fails, nodes j and k send update messages to their neighboring nodes as shown in Figure 9(b). In this example, node k is forced to report an infinite distance to j as nodes b and i have reported node k as part of their path to destination j. Node b processes node k's update and selects link (b, j) to destination j. This is because of step (2) of WRP which forces node b to purge any path to node j involving node k. Also, when i gets node k's update message, i updates its distance table entry through neighbor k and checks for the possible paths to destination j through any other neighboring nodes. Thus, a node examines the available paths through its other neighboring nodes and updates the distance and the routing table entries accordingly. This results in the selection of the link (i, J) to the destination j (Figure 9(c)). When node i receives neighbor b's update reporting an infinite distance, node i does not have to update its routing table as it already has correct path information (Figure 9(d)). Similarly, updates sent by node k reporting a distance of 11 to destination j will not affect the path information of nodes i and b. This illustrates how step (2) of WRP helps in the reduction of the information of temporary loops in the explicit paths.

3.1.3.6 Strengths and Weaknesses

Strength: The major advantage of WRP is that it reduces temporary looping by using the predecessor information to identify the route.

Weaknesses: In WRP, each node is required to maintain four routing tables. This can lead to substantial memory requirements, especially when the number of nodes in the network is large. Furthermore, if there are no recent packet transmissions from a given node, the WRP protocol requires the use of hello packets. The hello packets consume bandwidth and disallow a node to enter sleep mode.

3.1.4 Global State Routing

3.1.4.1 Protocol Description

Global State Routing (GSR) [59] is similar to DSDV. It takes the idea of link state routing but improves it by avoiding flooding of routing messages.

The network is modeled as an undirected graph G (V, E), where V is the number of nodes and E is the number of edges in the network. Nodes may move around and change their speed and direction independently. An undirected edge connecting two nodes is formed when the distance between two nodes is less than or equal to the transmission range of the wireless communication device.

In this algorithm, each mobile host has knowledge of full network topology and it maintains a Neighbor list, a Topology table, a Next Hop table and a Distance table. The neighbor list of a node contains the list of its neighbors. For each destination node, the Topology table contains the link state information as reported by the destination and the timestamp indicating the time the destination has generated this information. For each destination, the Next Hop table contains the next hop to which the packets for this destination must be forwarded. The Distance table contains the shortest distance to each destination node. Additionally, a weight function is used to compute the distance of a link. If min hop shortest path is the

objective, this weight function returns 1 if two nodes have direct connection. This weight function may also be replaced with other functions for routing with different metrics.

At the beginning, each node starts with an empty neighbor list and empty topology table. The node learns about learns about its neighbors by examining the sender field of each packet in the inbound queue. It is assumed that all nodes that can be heard are neighbors of that node.

Upon receiving a routing message, a node updates its topology table if the sequence number of the message is newer than the sequence number stored in the table. The concept of a sequence number is borrowed from DSDV and is used to distinguish stale links from new links.

The node then processes the received routing messages, which contain link state information broadcasted by its neighbors and builds the whole topology of the network, i.e. it builds up the network graph. Dijkstra's algorithm is used to compute the shortest path from this node to all other nodes in the network. The node then rebuilds the routing table based on the newly computed topology and broadcasts the new information to its neighbors. Since global topology is maintained at each node, preventing routing loops is easier.

3.1.4.2 Complexity comparison with LS and DBF

GSR and LS have the same memory complexity and computation as both maintain the topology of the whole network and use Dijkstra's algorithm to compute shortest path routes.

The packet complexity (the average number of routing packets exchanged by a node in each time slot) of LS can be high when mobility is high as LS transmits one short packet for each link update. On the other hand, both GRS and DBF transmit a fixed number of update tables using longer packets to optimize the MAC throughput.

The convergence time for GSR is superior to that for DBF. In fact, if shorter update interval is used, GSR can converge as fast as LS. The routing accuracy of GSR is comparable to ideal LS scheme and thus superior to the traditional DBF algorithm.

3.1.4.3 Strengths and Weaknesses

Strengths: It is superior to DBF (because of faster convergence and accurate routing info) and LS (because it does not use flooding). GSR avoids flooding problem for disconnects/reconnects by using periodic exchange of the entire topology map, greatly reducing the control message overhead.

Weaknesses: The weaknesses of GSR are the large size update messages and latency of the LS change propagation, which depends on the update period. This is where the fisheye technique comes to help, by reducing the size of update messages without seriously affecting routing accuracy.

3.1.5 Fisheye State Routing Algorithm

3.1.5.1 Protocol Description

Fisheye State Routing (FSR) [27, 29] is an improvement of GSR. The large size of update messages in GSR wastes a considerable amount of network bandwidth. The fisheye technique comes to help by reducing the size of update messages without seriously affecting routing accuracy.

FSR is an implicit hierarchical routing protocol. FSR uses the "fisheye" technique proposed by Kleinrock and Stevens [40], where the technique was used to reduce the size of information required to represent graphical data. The eye of a fish captures with high detail the pixels near the focal point. The detail decreases as the distance from the focal point increases. In routing, the fisheye approach translates to maintaining accurate distance and path quality information about the immediate neighborhood of a node, with progressively less detail as the distance increases.

Similar to Link State Routing (LS), FSR maintains a topology map at each node. In FSR, nodes maintain a link state table based on the up-to-date information received from neighboring nodes, and periodically

exchange it with their local neighbors only (no flooding). Through this exchange process, the table entries with larger sequence numbers replace the ones with smaller sequence numbers.

In a wireless environment, a radio link between mobile nodes may experience frequent disconnects and reconnects. The LS protocol releases a link state update for each such change, which floods the network and causes excessive overhead. FSR avoids this problem by using periodic, instead of event driven, exchange of the topology map, greatly reducing the control message overhead.

When the network size grows large, the update message could consume considerable amount of bandwidth, which depends on the update period. In order to reduce the size of update messages without seriously affecting routing accuracy, FSR uses the fisheye technique. Figure 10 illustrates the application of fisheye in a mobile, wireless network. The circles with different shades of grey define the fisheye scopes with respect to the center node (11). The fisheye scope is defined as the set of nodes that can be reached with a given number of hops. In our case, three scopes are shown for 1, 2 and >2 hops respectively. Nodes are color coded as black, gray and white accordingly. The number of levels and the radius of each scope will depend on the size of the network.

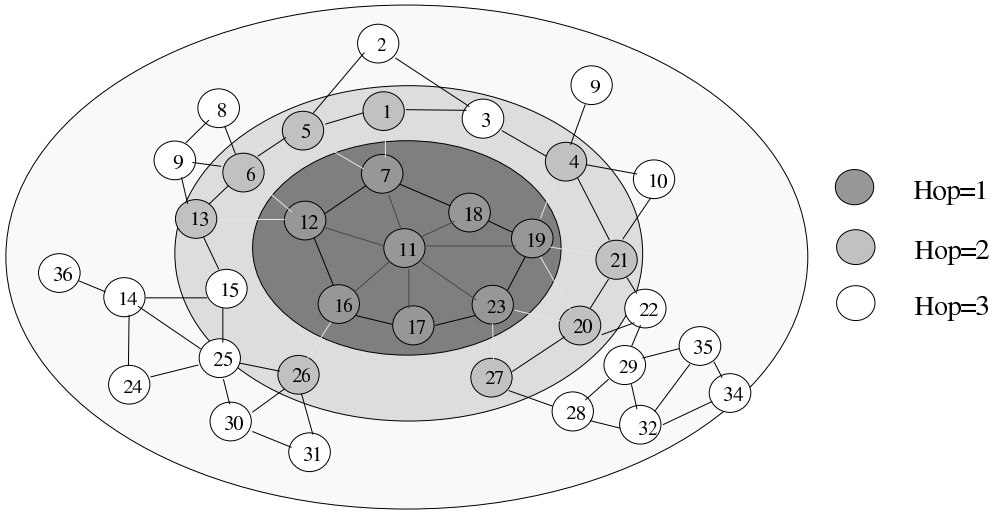


Figure 10. Scope of fisheye

Entries in routing table corresponding to nodes within the smallest scope are propagated to the neighbors with the highest frequency. Referring to Figure 11, entries in bold are exchanged most frequently. The rest of the entries are sent out with lower frequencies. As a result, a considerable fraction of link state entries is suppressed in a typical update, which reduces the message size. Thus by using different exchange periods for different entries in the routing table, routing update overhead is reduced. This strategy produces timely updates from near stations, but creates large latencies from stations afar.

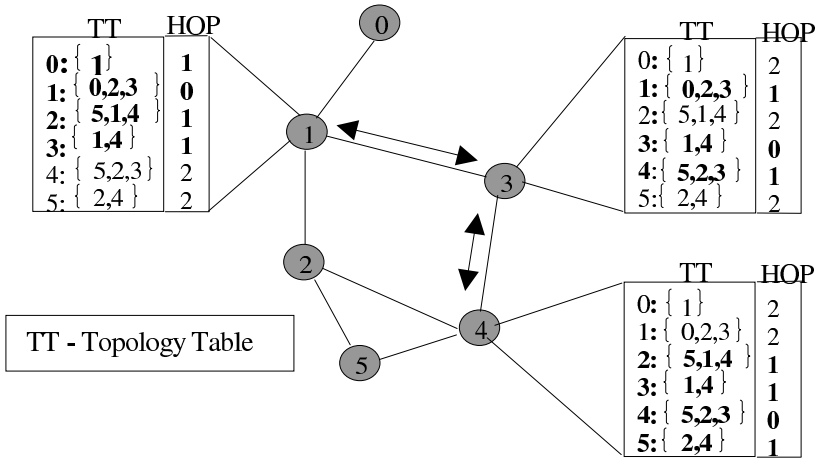


Figure 11. Message reduction using fisheye

Even though the nodes do not have accurate information about distant nodes, the packet is routed correctly because when a packet approaches its destination, it finds increasingly accurate routing instructions as it enters sectors with a higher refresh rate.

3.1.5.2 Complexity

In this section we compare the FSR with that of other three routing schemes (GSR, DBF and LS). Table 8 shows the comparison. FSR, GSR and LS have same memory complexity (MC) and computation complexity (CC). FSR and LS both maintain the topology for the whole network and use Dijkstra's algorithm to compute shortest path routes. Dijkstra's algorithm requires typically $O(N^2)$ steps to compute the shortest paths from the source to all destinations. DBF has complexity of O(N) for computing and memory, as it only keeps the distance information for each destination, and computes shortest paths in a distributed fashion.

Protocol	CC	MC	LO	CO	CT
FSR	$O(N^2)$	$O(N^2)$	$O(\sum_{i=1}^{L} \frac{n_i d}{w_i}) / I$	O (1)	O (D.I)
GSR	$O(N^2)$	$O(N^2)$	O (N.d) / I	O (1)	O (D.I)
LS	$O(N^2)$	$O(N^2)$	O (N.d) / I	O (N)	O (D)
DBF	O (N)	O (N)	O (N) / I	O (1)	O (N.I)

Table 8. Complexity comparison

GSR and LS have the same line over head (LO) but different from FSR. In LS, has similar accumulated data size for each link update, but its update interval 'I' may become extremely small when mobility increases.

In addition, as LS transmit one short packet for each link update, its control packet complexity (CO) can be as high as O(N) when the mobility is high. On the other hand, both FSR and DBF transmit a fixed number of update tables using longer packets to optimize the MAC throughput. Thus, CO=O(1). Finally, the convergence time for FSR is also superior than that for DBF .

3.1.5.3 Strengths and Weaknesses

Strengths: It provides lower latency for access to frequently used destinations. It has a lower control traffic overhead in dense traffic situation. Moreover, the protocol is simple and usage of up-to-date shortest route as well as robustness to the host mobility.

Weaknesses: Though compared to other flat table driven schemes (such as DSDV, GSR), it scales better, it still has scalability limitations due to the flat addressing scheme.

3.1.6 Hierarchical State Routing (HSR)

3.1.6.1 Cluster Formation and Election of Cluster heads

HSR is a hierarchical link state based routing protocol. It maintains a hierarchical topology. The network is partitioned into clusters and a cluster head is elected as in cluster-based algorithms.

When a node comes up, it searches for a cluster head in its neighborhood. If it is unable to find one, it forms a new cluster and elects itself as the cluster head. It then periodically broadcasts the message that it

is the cluster head. Any node now coming within the transmission range of this node listens to this message and accepts the broadcasting node to be its cluster head. All nodes within the transmission range of a cluster head belong to this cluster. That is, all the nodes in a cluster can communicate with a cluster head.

Initially, all nodes choose themselves to be cluster heads and form clusters. But then there will be "collisions" i.e. two cluster heads lying within the same cluster. So, the following method is adopted for maintaining a single cluster head. If two cluster heads come within transmission range of each other, then one cluster head challenges the other and one of the node gives up its cluster head position according to the lowest-id or highest-connectivity.

For a highly dynamic topology, frequent cluster head changes adversely affect the performance of other protocols. Thus Least Cluster Change (LCC) clustering algorithm is adopted to reduce the number of times the cluster head changes. According to LCC, the cluster head changes only in two conditions - one when two cluster heads come within range of each other, and other is when a node becomes disconnected from any other cluster. HSR provides both multilevel physical clustering and multilevel logical clustering.

3.1.6.2 Physical Multilevel Clustering

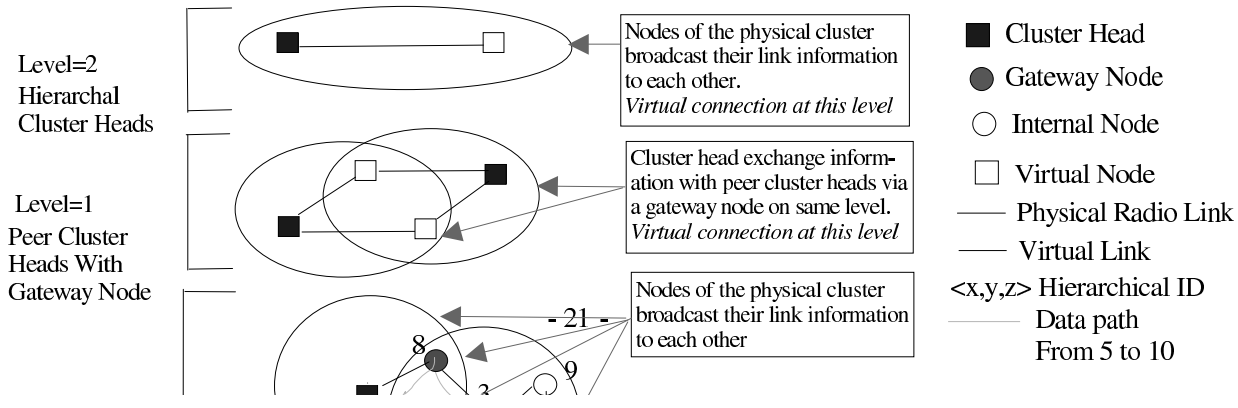
The clustering is based on geographical i.e. physical relationship between nodes. Within a physical cluster, each node monitors the link state to each neighbor and broadcasts it within the cluster.

The cluster head summarizes this information within its cluster and propagates it to neighbor cluster heads via gateways (nodes lying within more that one cluster). The knowledge of connectivity between neighbor cluster heads leads to the formation of level 1 clusters.

The cluster heads at lower levels again organize themselves into clusters and elect a cluster head and so a next higher level is created. Only, the lowest level consists of physical links. At higher levels only virtual links exist. These virtual links can be viewed as a "tunnel" implemented through lower level physical links. At higher levels, the LS information of the virtual link is exchanged. The nodes flood all the information to the lowest level so a physical node has the hierarchical topology information by which it can update its own HID. The *hierarchical ID (HID)* is defined as a sequence of the MAC addresses of the nodes from the top hierarchy to the node itself.

As shown in Figure 12 below the cluster heads 1,2,3 at level 0 organize into clusters at level 1, electing nodes 1 and 3 as cluster heads, making 2 a gateway node. The cluster heads 1 and 3 again organize into clusters, choosing 1 as cluster head. So, node 5, know about the hierarchical topology, assigns itself the HID <1,1,5>. Node 10 assigns itself the HID <3,3,10>. The routing is done by specifying the HID. Suppose node 5 sends a packet to node 10. The packet first goes into the topmost hierarchy as specified by the HID of the node 5 i.e. the packet goes from 5 to 1. Then it follows the HID of node 10 i.e. it goes to node 3 and then to node 10. This path that is just described is the virtual path that is followed in this hierarchical topology. The actual path that is followed by the packet consists of nodes 5,1,6,2,8,3,10.

Gateways nodes can communicate with multiple cluster heads and thus can be reached from the top hierarchy via multiple paths. Consequently, a gateway has multiple hierarchical addresses, similar to a router in the wired Internet, equipped with multiple subnet addresses.



3.1.6.2.1 Strengths and Weaknesses

Strengths: The advantage of this hierarchical address scheme is that each node can dynamically and locally update its own HID upon receiving the routing updates from the nodes higher up in the hierarchy.

Weaknesses: It has the cost of continuously updating the cluster hierarchy and the hierarchical address as nodes move. A continuously changing hierarchical address makes it difficult to locate and keep track of nodes.

3.1.6.3 Logical Partitions and Location (Membership) Management

The basic approach used is similar to mobile IP, except those here home agents are mobile too. The original framework of physical multilevel clustering is retained. Thus each node knows it's HID. In addition to HID addresses, nodes are assigned logical addresses of the type <subnet, host>. These addresses have a form similar to IP and in fact can be viewed as private IP addresses for the wireless network. Each subnet corresponds to a particular user group (e.g. students of the same class, tank battalion in the battlefield) and is associated with a home agent. Note that the IP sub network is a "virtual" sub network that spans several physical clusters.

Each member of a logical sub network knows the HID of its Home Agent (it is listed in the routing table). The corresponding nodes in the subnet must register with their Home Agent. The home agent, which is also a member of the subnet, manages membership of the corresponding nodes of the subnet. The registration is both periodic and event driven (e.g. whenever the member moves to a new cluster). The registered address is timed out and erased, if not refreshed. Since in most applications, the members of the same subnet move as a group, they tend to reside in neighboring clusters and thus registration overhead is modest.

The home agent advertises its HID (along with the subnet it represents) to the top hierarchy which is propagated down to all the nodes along with all the hierarchical information. So, each node has one to one mapping between the subnet id and HID of the home agent.

When a source wants to send a packet to a destination about which it knows the IP address, it first extracts the subnet address field from it. From its internal list or from the top hierarchy, it obtains the hierarchical address of the corresponding home agent. It then sends the packet to the home agent using that HID. The home agent finds the registered address (HID) from the host ID (in the IP address) and delivers packet to the destination. Once source and destination have learned each other's addresses, packets can be delivered

directly without involving the Home Agent. Since the logical address is used for routing, it is adaptable to topology changes.

3.1.6.3.1 Strengths and Weaknesses

Strengths: Since in most applications, the members of the same subnet move as a group (e.g., tanks in a battalion), they tend to reside in neighboring clusters. This reduces the registration overhead.

Weaknesses: Because of Home Agent, the protocol complexity is higher than flat LS routing.

3.2 Reactive (On-demand) Protocols

These protocols take a lazy approach to routing. In contrast to table-driven routing protocols all up-to-date routes are not maintained at every node, instead the routes are created as and when required. When a node requires a route to a destination, it invokes the route discovery mechanisms to find the path to the destination. This process is completed once a route is found or all possible route permutations have been examined. Once a route has been established, it is maintained by a route maintenance procedure until either the destination becomes inaccessible along every path from the source or until the route is no longer desired. This section discusses two on-demand routing protocols.

3.2.1 Ad-Hoc On Demand Distance Vector Protocol (AODV)

The Ad-Hoc On Demand Distance Vector (AODV) routing algorithm is a routing protocol designed for ad-hoc mobile networks. It offers quick adaptation to dynamic link conditions, low processing and memory overhead. It uses destination sequence numbers to ensure loop freedom at all times, solving problems (such as "counting to infinity") associated with classical distance vector protocols.

3.2.1.1 AODV Properties

The Ad-Hoc On Demand Distance Vector (AODV) routing algorithm is an on demand algorithm, meaning that it builds routes between nodes only as needed by source nodes and are maintained only as long as they are necessary. AODV is loop free at all times, even while repairing broken links [8]. This loop freedom is accomplished through the use of sequence numbers. Every node maintains its own monotonically increasing sequence number, which it increases each time it learns of a change in the topology of its neighborhood. This sequence number ensures that the most recent route is selected whenever route discovery is executed. AODV is able to provide unicast, multicast, and broadcast communication ability. In addition, AODV forms trees, which connect multicast group members. The trees are composed of the group members and the nodes needed to connect the members. Lastly, it is expected that continued improvements to the basic algorithm (e.g., for QoS applications, for client-server discovery, or for utilizing asymmetric routing paths) will benefit both unicast and multicast data transmission.

AODV currently utilizes only symmetric links between neighboring nodes. Routing tables are used by AODV to store pertinent routing information and utilized both unicast routes table and multicast routes table. It is able to maintain both unicast and multicast routes even for nodes in constant movement. It also provides for the quick deletion of invalid routes through the use of a special route error message. AODV responds to topological changes that affect active routes in a quick and timely manner. It builds routes with only a small amount of overhead from routing control messages and no additional network overhead. Finally, AODV does not place any additional overhead on data packets because it does not utilize source routing.

3.2.1.2 Unicast Route Establishment

Route discovery with AODV is purely on demand and follows a routes request/route reply discovery cycle. Requests are sent using a Route Request (RREQ) message. Information enabling the creation of a route is sent back in a Route Reply (RREP) message.

3.2.1.2.1 Route Discovery

When a source node desires to send a packet to some destination node and does not already have a valid route to that destination, it initiates a *route discovery* process. To begin such a process, the source node creates a RREQ packet. In addition to the source node's IP address, current sequence number, and broadcast ID, the RREQ also contains the most recent sequence number for the destination of which the source node is aware. After creating the RREQ, the source node broadcasts the packet to its neighbors, which then forward the request to their neighbors, and so on, until either the destination or an intermediate node with a "fresh enough" route to the destination is located and then sets a timer to wait.

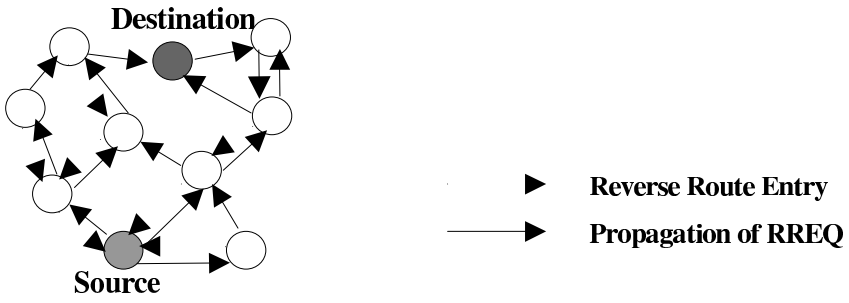


Figure 13: Propagation of RREQ throughout the Network

for the reply. Nodes keep track of the RREQ's source IP address and broadcast ID. If they receive a RREQ, which they already processed, they discard the RREQ and do not forward it

To process this RREQ, the node sets up a *reverse route* entry for the source node in its route table which contains the source node's IP address and sequence number as well as the number of hops to the source node and the IP address of the neighbor from which the RREQ was received. Figure 13 indicates the propagation of RREQs across the network as well as the formation of the reverse route entries at each of the network nodes. Associated with each route entry is a route timer which will cause the deletion of the entry if it is not used within the specified lifetime.

A node receiving the RREQ may send a route reply (RREP) if it is either the destination or if it has a route to the destination with corresponding sequence number greater than or equal to that contained in the RREQ. If this is the case, it unicasts a RREP back to the source. Otherwise, it rebroadcasts the RREQ.

3.2.1.2.2 Expending Ring Search

Each time a node initiates route discovery for some new destination, it must broadcast a RREQ across the network. For a small network, the impact of this flooding is minimal. However, for a large network the impact may become increasingly detrimental. To control network-wide broadcasts of RREQs, the source node should use an expanding ring search technique, which allows a search of increasingly larger areas of the network if a route to the destination is not found. To use the expanding ring search, the source node sets the Time to Live (TTL) value of the RREQ to an initial **ttl_start** value. If no reply is received within the discovery period, the next RREQ is broadcasts with a TTL value increased by an increment value. This process of increasing the TTL value continues until a threshold value is reached

3.2.1.2.3 Forward Path Setup

Eventually, a RREQ will reach a node (possibly the destination itself) that possesses a current route to the destination. The receiving node first checks that the RREQ was received over a bi-directional link. If an intermediate node has a route entry for the desired destination, it determines whether the route is current by comparing the destination sequence number in its own route entry to the destination sequence number in the RREQ. If the RREQ sequence number for the destination is greater than that recorded by the intermediate node, the intermediate node must not use its recorded route to respond to the RREQ. Instead, it rebroadcasts the RREQ. The intermediate node can reply only when it has a route with a sequence number that is greater than or equal to that contained in the RREQ. If it does have a current route to the destination, and if the RREQ has not been processed previously, the node then unicast a route reply packet (RREP) back to its neighbor from which it received the RREQ.

By the time a broadcast packet arrives at a node that can supply a route to the destination, a reverse path has been established to the source of the RREQ. As the RREP travels back to the source, each node along the path sets up a forward pointer to the node from which the RREP came, updates its timeout information for route entries to the source and destination, and records the latest destination sequence number for the requested destination. Figure 14 indicates the forward path setup as the RREP travels from the destination to the source node. Nodes that are not along the path determined by the RREP will timeout after ACTIVE_ROUTE_TIMEOUT (3000/msec) and will be deleted.

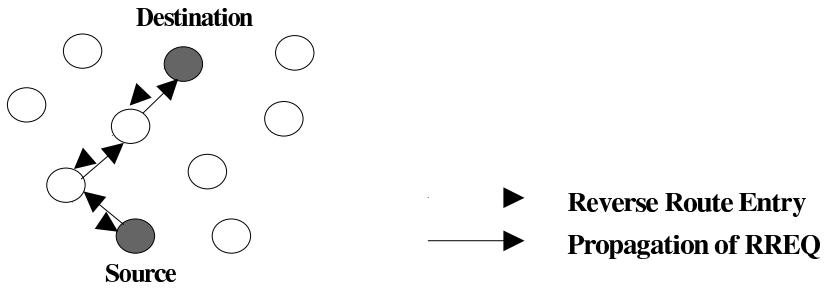


Figure 14. Route Determination from Source to Destination

If node receives multiple RREP, propagates first, and additional ones only if they contains either a greater destination sequence number than previous RREP or the same destination sequence number with a smaller hop count.

3.2.1.2.4 Route Table Management

AODV needs to keep track of the following information for each route table entry:

- Destination IP Address: IP address for the destination node.
- Destination Sequence Number: Sequence number for this destination.
- Hop Count: Number of hops to the destination.
- Next Hop: The neighbor, which has been designated to forward packets to the destination for this route entry.
- Lifetime: The time for which the route is considered valid.
- Active neighbor list: Neighbor nodes that are actively using this route entry.
- Request buffer: Makes sure that a request is only processed once.

3.2.1.2.5 Route Maintenance

Once a route has been discovered for a given source/destination pair, it is maintained as long as needed by the source node. Movement of nodes within the ad hoc network affects only the routes containing those nodes; such a path is called an active path. Movement not along an active path does not trigger any protocol action. If the source node moves during an active session, it can reinitiate route discovery to establish a new route to the destination. When either the destination or some intermediate node moves, however, a Route Error (RERR) message is sent to the affected source node. This RERR is initiated by the node upstream of the break (i.e., closer to the source nodes). It lists each of the destinations that are now unreachable because of the loss of the link. If the node upstream of the break has one or more nodes listed as a precursor node for the destination (implying that one or more nodes route through it in order to reach the destination), it broadcasts the RERR to these neighbors. When the neighbors receive the RERR, they mark their route to the destination as invalid by setting the distance to the destination equal to infinity and in turn propagate the RERR to their precursor nodes, if any such nodes are listed for the destinations in their route tables. When a source node receives the RERR it can reinitiate route discovery if the route is still needed.

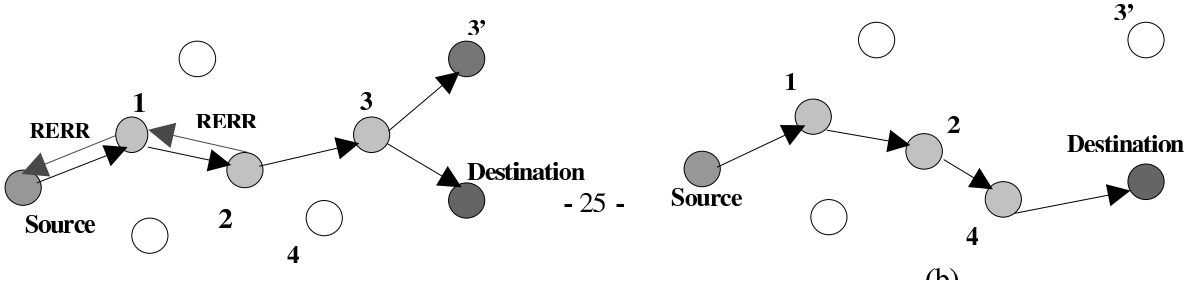


Figure 15 illustrates the route maintenance procedure. In Figure 15, the original path from the source to the destination is through nodes 1, 2, and 3. Node 3 then moves to location 3', causing a break in connectivity with node 2. Node 2 notices this break and send a RERR to node 1. Node 1 marks this route as invalid and then forward the RERR to the source. On receiving the RERR, the source node determines that it still needs the route, and so it reinitiates route discovery. Figure 15 (b) shows the new route found through node 4.

3.2.1.2.6 Local Connectivity Management

Neighborhood information is obtained from broadcasts sent by neighboring nodes. Whenever a node receives a broadcast from its neighbor, it updates its local connectivity information to ensure that it includes this neighbor. If there is no entry for that neighbor already in the table, the node creates one. In the event that a node has not sent any packets to all of its active downstream neighbor within **hello_interval**, it broadcasts to its neighbors a Hello message (a special unsolicited RREP), containing its identity and current sequence number. It is prevented from being rebroadcast outside the neighborhood of the node because it contains a TTL value of 1. The failure to receive any transmissions from a neighbor in the time defined by the periodic transmission of several Hello messages is an indication that the local connectivity has changed and that the route information for this neighbor should be updated.

3.2.1.2.7 Actions after Reboot

After reboot a node will have lost its prior sequence number as well as its last known sequence numbers for various other destinations. Since neighboring nodes may be using this node as an active next hop, it can create routing loops**.** To prevent this possibility, each node on reboot waits for **delete_period**, during which it does not respond to any routing packets. However, if it receives a data packet, it broadcasts a RERR and resets the waiting timer (lifetime) to expire after the current time plus **dalete_period.**

3.2.1.3 Broadcast

AODV specifies behavior for transmitting broadcasts. When a node wishes to generate a broadcast, it sends the broadcast packet to well-known broadcast address 255.255.255.255.

When a node receives a packet broadcast to address 255.255.255.255, it notes the source IP address, the IP ident value, and the fragment effect of the packet's IP header. It then checks its broadcast list entries to determine whether the packet has already been received and thus whether it has already been retransmitted. If there is no such matching entry, the node processes and retransmits the broadcast packet. Otherwise, the silently discards the packet

3.2.1.4 Security Considerations

Currently, AODV does not specify any special security measures. Route protocols, however, are prime targets for impersonation attacks. These attacks include transmitting RREPs with false routing information and denial of service attacks through the repeated broadcast of RREQ messages. Additionally, wireless transmission is inherently insecure. Packets are received by anyone within transmission range, and if they are not encrypted, they can also be read by anyone. To protect against these attacks, authentication techniques involving generation of unforgeable and cryptographically strong message digests or digital signatures can be used. It is expected that, in environments where security is an issue, that IPSec authentication headers will be deployed along with the necessary key management to distribute keys to the members of the ad hoc network using AODV.

3.2.1.5 Optimizations And Enhancements

Through various simulations, AODV has shown excellent performance in various network scenarios. However, there are still ways to improve it as well as to enhance it so that it can provide additional services and be more versatile. The following sections describe methods for optimizing and enhancing AODV.

3.2.1.5.1 Quality of Service

AODV defines extensions that can be used to request certain quality of service parameters - in particular, Maximum Delay and Minimum Bandwidth - which cab be appended to a RREQ. To respond to a RREQ with such extensions, a node must be able to satisfy the indicated quality of service constraints.

If after the establishment of such a route, any node along the path detects that it can no longer maintain the requested quality of service parameters, it must send an ICMP **QOS_LOST** message back to the node that originally requested them.

No additional processing is needed for the Minimum Bandwidth extension, because intermediate nodes rebroadcast the RREQ message containing it. For the Maximum Delay extension, each intermediate node must subtract its own characteristic value for adding delay to the handling of future data packets. This characteristic value depends on measured processing time as well as time-varying queue lengths. It is expected that queue lengths for incoming data packets will be fairly volatile, so a conservative reported value is prepared. Moreover, a forwarding node wishing to offer such delay assurances may be required to reduce queue delays for incoming packets from the requesting node. Further work may be needed to allow the forwarding node to distinguish among flows of various types from the same IP source node.

3.2.1.5.2 Subnet Routing

In case of subnet routing, a route to any one of the nodes in the collection is nearly as good as a route to any other node, plus or minus one hop. Thus, route information to all of the nodes can be summarized by a single route table entry, and route aggregation is possible.

To work with AODV, routes to the subnet have to be assigned a destination sequence number just as multicast groups are assigned a sequence number. For subnet, all that is needed is that one of the nodes on the subnet takes responsibility for creating and managing the sequence number. If there is router on the subnet already that node is the logical choice, if not, some other node has to be assigned this function as well as the function of forwarding traffic for other nodes on the subnet. The node managing the sequence number is called the subnet leader, and it must be consider the default router for all subnet nodes.

Nodes on the subnet that receive RREQ have to forward them to the subnet leader. The subnet leader creates a reverse route to its subnet nodes in the same way as to any other node in the network. RREP messages through any node on the subnet must be sent back to the source through the subnet leader. In subnet, though all route establishments are on demand, no periodic route advertisement is required.

3.2.1.6 Complexities

AODV comparisons with some existing routing algorithms in terms of time and communication complexities are made, as shown in Table 9.

Type of Routing Protocol	Time Complexity	Communication
AODV (initialization)	*O (2d)*	*O (2N)*
AODV (postfailure)	*O (2d)*	*O (2N)*
DSR (initialization)	O (2d)	O (2N)
DSR (postfailure)	O (2d) or o (cache hit)	O (2N)
TORA (initialization)	O (2d)	O (2N)
TORA (postfailure)	O (2d)	O (2x)
ABR (initialization)	O (d+z)	O (N+y)
ABR (postfailure)	O (l+z)	O (x+y)
SSR (initialization)	O (d+z)	O (N+y)
SSR (postfailure)	O (l+z)	O (x+y)

Table 9. Complexity Comparison of ABR with Some Existing Routing Protocols

3.2.1.7 Strengths and Weaknesses

Strengths: It uses bandwidth efficiently (by minimizing the network load for control and data traffic), is responsive to changes in topology, is scalable and ensures loop free routing. AODV is particularly novel in its support for multicast routes. AODV is the only protocol to date that is capable of discovery of both unicast and multicast routes. All other current protocols provide one of unicast or multicast route discovery, but not both.

Weaknesses: Simulation results [20] indicates that for longer packets, resulting in more collision and longer queue lengths at intermediate nodes, which in turn delays RREQ and RREP messages and increases route acquisition latency.

3.2.2 Dynamic Source Routing (DRS)

3.2.2.1 Protocol Description

Dynamic Source Routing (DSR) [17,16,33] also belongs to the class of on demand protocols and allows node to dynamically discover a route across multiple network hops to any destination. Source routing means that each packet in its header carries the complete ordered list of nodes through which the packet must pass. DSR uses no periodic routing messages (e.g. no router advertisements), thereby reducing network bandwidth overhead, conserving battery power and avoiding large routing updates throughout the ad-hoc network. Instead DSR relies on support from the MAC layer (the MAC layer should inform the routing protocol about the link failure). The two basic models of operation in DSR are route discovery and route maintenance.

3.2.2.2 Route Discovery

Route discovery is the mechanism whereby a node S wishing to send a packet to some destination D, obtains a source route to D. To establish a route, the source S broadcasts a Route Request (RREQ) packet with a unique request ID that may be received by those hosts within wireless transmission range of it. When this request message reaches the destination or a node that has route information to the destination, it sends a Route Reply (RREP) message containing path information back to the source. The "route cache" maintained at each node records routes the node has learned and overheard over time to reduce overhead generated by a route discovery phase.

When a node receives a route request packet, it processes the request according to the following steps:

- If the pair (initiator address, request ID) for this route request is found in this host's list of recently seen requests, then discard the route request packet and do not process it further.
- Otherwise, if this host's address is already listed in the route record in the request, then discard the route request packet and do not process it further.
- Otherwise, if node is the intended target, the route record is complete, reply with route reply.
- Otherwise, append node to route record in the route request packet, and re-broadcast.

3.2.2.3 Route Maintenance

Route maintenance is the mechanism by which a packet sender S detects if the network topology has changed so that it can no longer use its route to the destination D. This might happen because a host, listed in a source route, moves out of wireless transmission range or it turned off making the route unusable. A failed link is detected by either actively monitoring acknowledgements or passively by running in promiscuous mode, overhearing that a packet is forwarded by a neighboring node.

When route maintenance detects a problem with a route in use, a Route Error (RERR) packet is sent back to the source node S. When this RERR packet is received, the hop in error is removed from this hosts route cache, and all routes that contain this hop are truncated at this point. Figure 16 shows the basic operation of DSR.

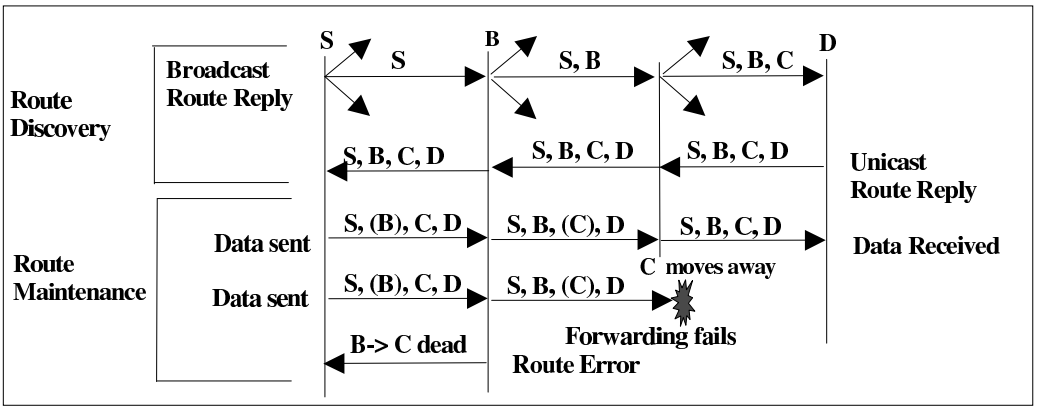


Figure 16. Basic operation of the DSR

3.2.2.4 Optimizations

To improve the performance and reduce overhead, a few optimizations can be achieved in DSR. Some of the optimizations are:

- **Full Use of the Route Cache:** The data in a host's route may be stored in any format, but the active routes, in its cache, form a tree of routes, rooted at this host, to other hosts in the network.

  A host can add entries to its route cache any time it learns a new route. When a host forwards a data packet as an intermediate hop on the route in that packet, the forwarding host is to observe the entire route in the packet. If a host forwards a route reply packet, it can also add the route information from the route record being returned in that route reply, to its own route cache. Since all wireless network transmissions are inherently broadcast, a host may be able to configure its network interface into promiscuous receive mode, and can then add to its route cache the route information from any data or route reply packet it can overhear.

  A host may use its route cache to avoid propagating a route request packet received from another host. For example, if mobile host F needs to send a packet to mobile host D, it will initiate a route discovery and broadcast a route request packet. If this broadcast is received by A, A can simply return a route reply packet to F containing the complete route to D consisting of the sequence of hops A, B, C, and D.

  A particular problem can occur, however, when several mobile hosts receive the initiator's broadcast of the route request packet, and all reply based on routes found in their route caches. When more then two mobile hosts in this example are involved, these simultaneous replies from the mobile hosts receiving the broadcasts may create packet collisions among some or all of these replies and may cause local congestion in the wireless network.

  To avoid this problem, each host delays slightly (delay period d= H*(h-1+r), where h is the length in number of network hops for the route to be returned in this host's reply, r is a random number between 0 and 1, and H is a small constant delay to be introduced per hop before replying from the cache. Using a delay period proportional to no. of hops to the destination increases the probability that the source receives the shortest route first.

  Another problem that can occur when hosts reply on route requests from their cache, is the formation of a loop in the route sent in the route reply packet. In order to avoid this problem, if a host receives a route request and is not the target of the request but could reply from its cache, the host instead discards the request if the route in its reply would contain a loop.

Finally, source nodes can generate non-propagating route requests by setting the TTL to zero hops. This guarantees that only neighbors receive the message and prohibits them from broadcasting it further. This technique allows the source node to query its neighbors cache for a route to the destination. If a reply is not received by a given time out period, a regular route request message is sent. This technique has the potential to reduce the protocol overhead, increase energy-efficiency, and reduce the congestion on limited capacity links.

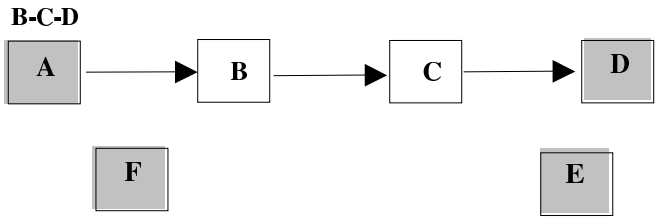


Figure 17. An example ad hoc network illustrating use of the route cache

- **Piggybacking on route request messages:** DSR can also piggyback data on the route request message in an effort to reduce discovery latency. If a node which is not the destination replies with a cached route to the destination, that node must be capable of building a data packet and forwarding it to the destination so that no information will be lost. This technique has the potential to reduce latency, the most significant drawback to on-demand protocols.
- **Reflecting Shorter Routes:** While two hosts are communicating with each other using cached routes, it is desirable for hosts to begin using shorter routes if the hosts move sufficiently close together. In many cases, the basic route maintenance procedure is sufficient to accomplish this, since if one of the hosts moves enough to allow the route to be shortened, it will likely be out of transmission range of the first hop on the existing route.

  An improvement to this method of reflecting shorter routes is possible if hosts operate their network interfaces in promiscuous receive mode. Suppose somewhere in the forwarding of a packet, mobile host B transmits a packet to C, with D being the next hop after C in the route in the packet, as illustrate in Figure 18. If D receives this packet, it can examine the packet header to see that the

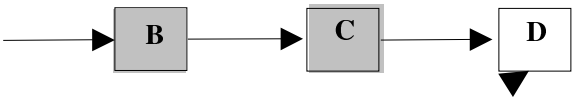


Figure 18. Mobile host D notices that the route can be shortened

  packet reached it from B in one hop rather than two as intended by the route in the packet. In this case, D may infer that route may be shortened to exclude the intermediate hop through C. D then sends an unsolicited route reply packet to the original sender of the packet, informing it that it can now reach D in one hop from B.
- **Handling Network Partitions:** One common error condition that must be handled in an ad hoc network is the case in which the network effectively becomes partitioned. That is, two hosts that wish to communicate are not within transmission range of each other, and there are not enough other mobile hosts between them to form a sequence of hops through which they can forward packets. If a new route discovery were to be initiated for each packet sent by a host in this situation, a large number of unproductive Route Request packets would be propagated throughout the subnet of the ad hoc network reachable from this host. In order to reduce the overhead from such route discoveries, an exponential backoff algorithm to be used to limit the rate which new route discoveries may be initiated from any host for the same target.

- **Eavesdrop on Route Error Packets (Promiscuous Mode):** Another optimization possible to improve the handling of errors is to use promiscuous receiving mode to allow host to eavesdrop on Route Error packets being sent to other hosts. Since a Route Error packet names both ends of the route hop causing the error, any host receiving the error packet can update its route cache to reflect the fact that the two hosts indicated in the packet can no longer directly communicate. A host receiving a Route Error packet can simply search its route cache for any routes using this hop, and for each such hop found, the route is truncated at this hop. A last optimization to improve the handling of errors is to support the caching of "negative" information in a host's route cache.
- **Symmetrical link support:** Although DSR supports asymmetrical links, it can be configured so that only bi-directional links are used. If all route request messages are required to return to the source node on the reverse path by which they came, route request messages will not be received over unidirectional links.
- **Salvaging:** In addition, DSR is capable of salvaging. Salvaging occurs when an intermediate node on the destination path realizes the next hop is unreachable. If an alternate route is cached, it is substituted for the broken link. The intermediate node, however, is still required to send a route error message to the source.

3.2.2.5 Integration with the Internet

Maltz et al. extended the mechanisms of Route Discovery Route and Maintenance to support communication between nodes inside the ad hoc network and those outside in the greater Internet. So that each node in the ad hoc network maintains a constant identity as it communicates with nodes inside and outside the network, we require that each node choose a single IP address, called its home address, by which it is known to all other nodes. This notion of a home address is identical to that defined by Mobile IP [4] As in Mobile IP, each node is configured with its home address and uses this address as the IP source address for all of the packets it sends.

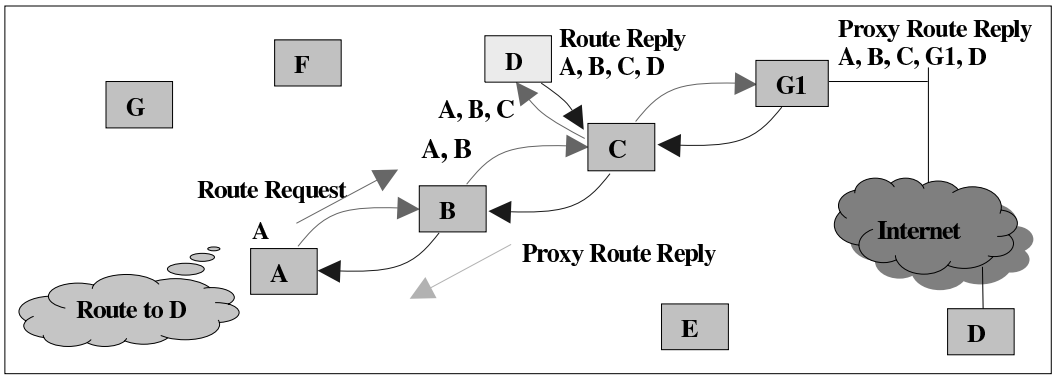


Figure 19. Route Request for a node not in the ad hoc network being answered by the foreign agent

Figure 19 illustrate node A inside the ad hoc network discovering a route to a node D outside the network. As the Route Request from A targeting D propagates, it is eventually received by the gateway node G1, which consults its routing table. If it believes D is reachable outside the ad hoc network, it sends a Route Reply listing itself as the second-to-last node in the route, and marking the reply such that A will recognize it as a proxy reply. If the target node D actually is inside the ad hoc network, node A will receive a Route Reply from both G1 and D. Since A can distinguish which replies are proxy replies, it can prefer the direct route when sending packet to D

3.2.2.6 Integration with Mobile IP

DSR supports the seamless interoperation between an ad hoc network and the Internet, allowing packets to be routed transparently from the ad hoc network to nodes in the Internet and from the Internet to nodes in the ad hoc network [31] Since node MN must be able to participate in different IP subnets depending on its current location, it uses Mobile IP to connect to the Internet. Figure 20 shows how MN is homed in a subnet not belonging to the ad hoc network, but MN has wandered in range of the ad hoc network and that node G1 is a gateway between the ad hoc network and the Internet that provides Mobile IP foreign agent services.

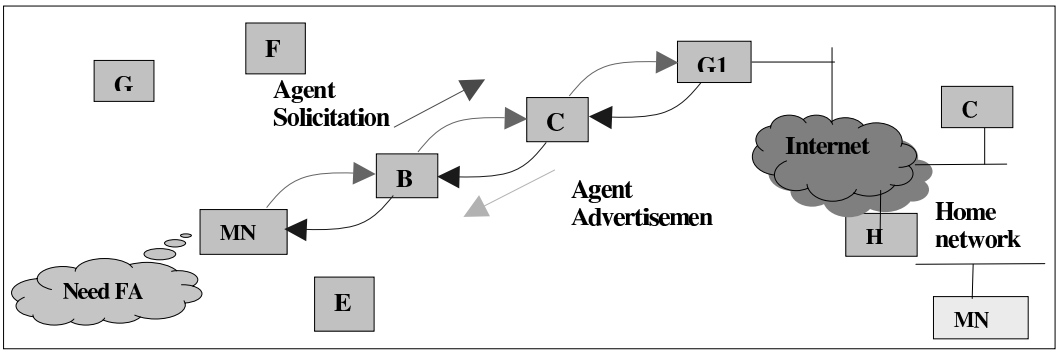


Figure 20. The roaming node registering with a foreign agent located on G1 in the ad hoc network

The mobile node (MN) will typically keep its network interface in promiscuous receive mode and so will know that it has entered a DSR network when it overhears DSR packets like Route Request, Route Reply or data packets with DSR source routes on them.

If MN decides its best connectivity would be via the ad hoc network, it transmits a Mobile IP AGENT SOLICITATION piggybacked on a Route Request targeting the IP limited broadcast address (255.255.255.255). This allows the SOLICITATION to propagate over multiple hops through the ad hoc networks, though gateways will not propagate it between subnets. When the foreign agent at G1 receives the SOLICITATION, it will reply with an AGENT ADVERTISEMENT, allowing MN to register itself with this foreign agent and with its home agent as a Mobile IP mobile node visiting the ad hoc network. Once the registration is complete, the mobile node's home agent will use Mobile IP to tunnel packets destined for the mobile node MN to the foreign agent at G1, and G1 will deliver the packets locally to the mobile node using DSR.

3.2.2.7 Multicast Routing with DSR

DSR does not currently support true multicast routing, but it does support an approximation that is sufficient in many network contexts. Though an extension of the Route Discovery mechanism, DSR supports the controlled flooding of a data packet to all nodes in the ad hoc network that are within some specified number of hops of the originator. These nodes may then apply destination address filtering (e.g., in software) to limit the packet to those nodes subscribed to the packet's indicated multicast destination address. Even though this mechanism does not support pruning of the broadcast tree to conserve network resources, it can be used to distribute information to all nodes in the ad hoc network subscribed to the destination multicast address. This mechanism may also be useful for sending application-level packets to all nodes in a limited range around the sender.

In this form of multicasting, an application on a DSR node sends a packet to a multicast destination address and DRS piggybacks the data from the packet inside a Route Request targeted at the multicast address. The normal Route Request propagation scheme results in this packets being efficiently distributed to all nodes in the network within the specified hop count (TTL) of the originator. After forwarding the packet as defined for Route Discovery, each receiving node then individually examines its

destination address and discards the packet if it is destined for a multicast address to which this node is not subscribed.

3.2.2.8 Complexities

The time complexity and communication complexities of DSR same as described in AODV.

3.2.2.9 Strengths and Weaknesses

Strengths:

- DSR uses the key concept of source routing. Intermediate nodes do not need to maintain up-to-date routing information in order to route the packets they forward. There is also no need for periodic routing advertisement messages, which will lead to reduced network bandwidth overhead, particularly during periods when little or no significant host movement is taking place. Battery power is also conserved on the mobile hosts, both by not sending the advertisements and by not needing to receive them, a host could go to sleep instead.
- The protocol has the advantage of learning routes by scanning for information in packets that are received. A route from A to C through B means that A learns the route to C, but also that it will learn the route to B. The source route will also mean that B learns the route to A and C and that C learns the route to A and B. This form of active learning is very good and reduces overhead in the network.
- No assumption of link symmetry is required
- Possible to react to changes faster than state-based or distance-vector based protocols.

Weaknesses:

- DSR may have scalability problems [61]. As the network becomes larger, control packets and message packets also become larger since they need to carry addresses for every node in the path. This may be a problem since ad-hoc networks have limited available bandwidth.
- Running the interfaces in promiscuous mode is a serious security issue, since the address filtering of the interface is turned off and all packets are scanned for information. A potential intruder could listen to all packets and scan them for useful information such as passwords and credit card numbers. Applications have to provide the security by encrypting their data packets before transmission. The routing protocols are prime targets for impersonation attacks and must therefore also be encrypted.
- Since DSR runs in promiscuous mode, it consumed high battery power.

3.2.3 Temporally-Ordered Routing Algorithm

3.2.3.1 Protocol Description

Temporally-Ordered Routing Algorithm (TORA) [60, 61, 62] is a highly adaptive, loop-free, scalable distributed routing protocol based on a "link reversal" algorithm and aims to build a directed cyclic graph rooted at the destination. The key design concept of TORA is to discover routes on demand, provide multiple routes to a destination, establish routes quickly, and minimize communication overhead by localizing the reaction to topological changes when possible. Route optimality (shortest-path routing) is considered of secondary importance, and longer routes are often used to avoid the overhead of discovering newer routes. It is also not necessary (nor desirable) to maintain routes between every source/destination pair at all times. The protocol performs four basic functions: creating routes, maintaining routes, erasing routes, and optimizing routes. TORA accomplishes these four functions through the use of three distinct control packets: query (QRY), update (UPD), and clear (CLR).

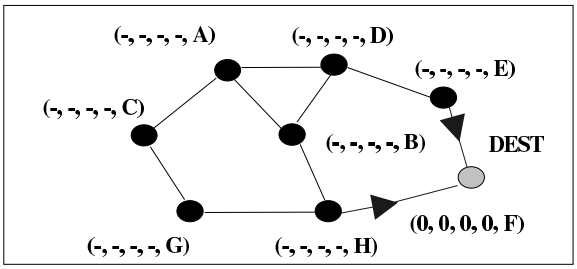
Each node has a quintuple associated with it- <logical time of link failure>, <the unique ID of the node, which defined the new reference level>, <a reflection indicator bit>, <an integer instrumental in propagation of new reference level>, <unique ID of the node itself>. The first three values define a

reference level while the last two define a delta with respect to a reference level, thus all the five values define the "height" of node.

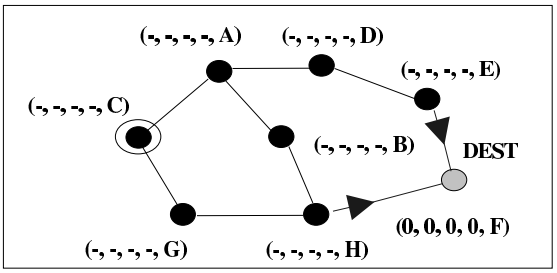
For the link (i, j) node i is said to be 'upstream' and j to be 'downstream' if "height" of node i is higher than that of j. Each node maintains only height and link state information of its neighbors. Initially, the height of each node is assigned to be null except that of the destination, which is assigned zero height so that it is at lowest level of 'stream'. If the height of any node is null, then any neighbor's height which is not NULL is considered 'lower'.

3.2.3.1.1 Creating Routes

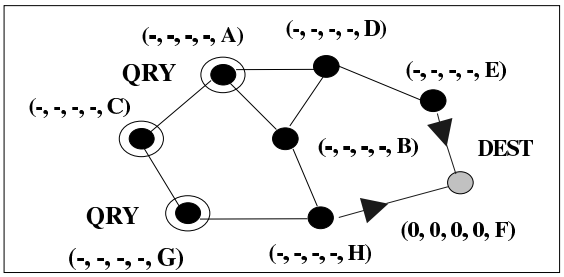
When a node needs a route to a particular destination, it broadcasts a Route Query (QRY) packet containing the address of the destination. This packet propagates through the network until it reaches either the destination, or an intermediate node having a route to the destination.



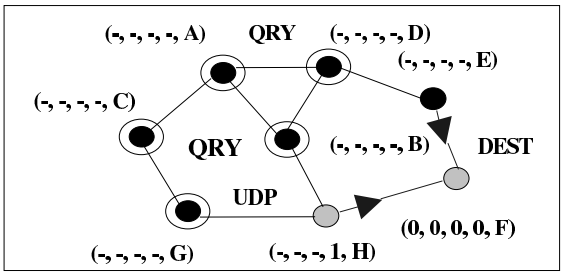
(a) Examples of creating routes process.
Initially, height of each node is null, except destination. So nodes E and H have their edges directed towards it



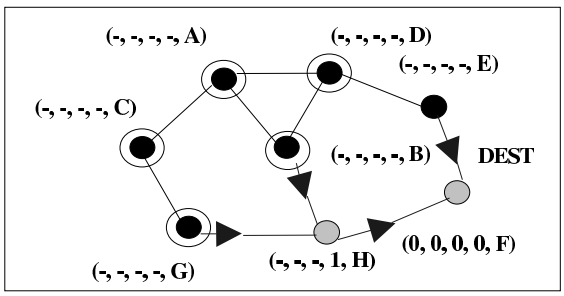
(b) Node C initiates QRY.
The circle around C (in the above figure) indicates that C is still looking for the route to DEST. It is implemented by setting a flag.



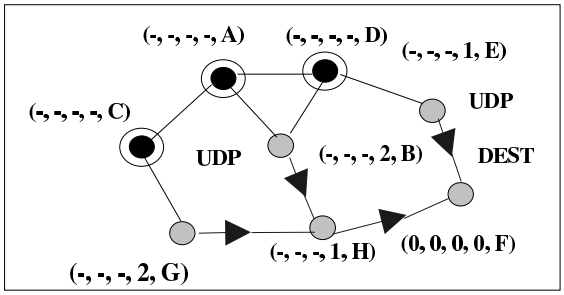
(c) Node A and G propagate QRY.
A and G receive the Query and since they do not have any downstream link/path to the DEST, they forward the Query packet.



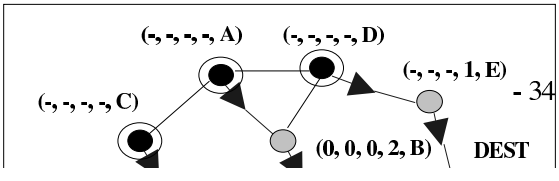
(d) Nodes B and D propagate QRY, node H generate UPD.
The update message consists of the height of the node propagating the message.

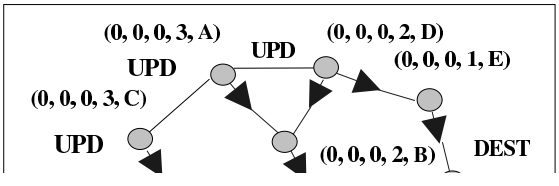


(e) Nodes B and G receive the update message and set their links directed towards H



(f) Nodes B and G propagate UPD and node E generate UPD.
The circle around B and G are removed since they now have a path to DEST.

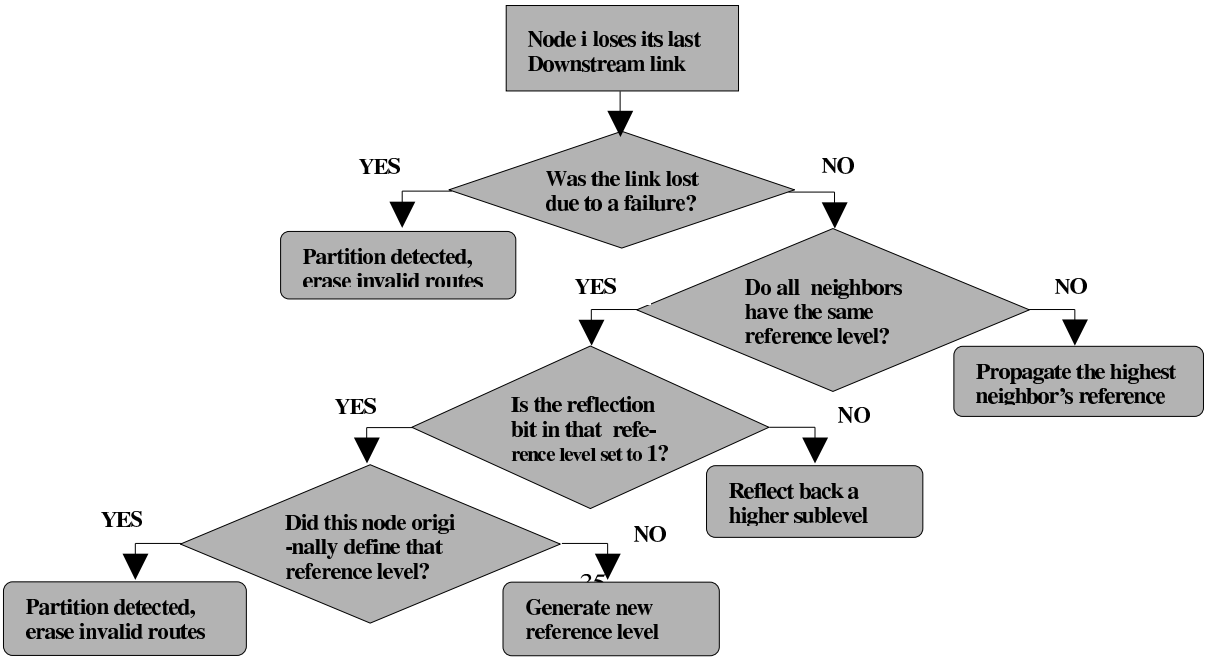


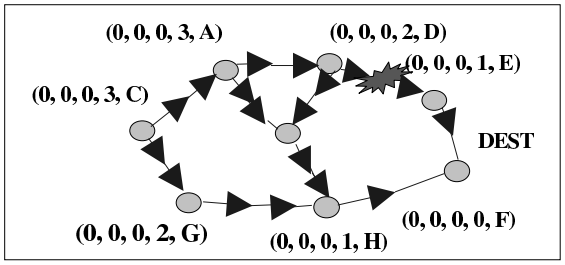


The recipient of the query packet then broadcasts an Update (UPD) packet listing its height with respect to the destination (if the recipient is the destination, this height is 0). As this packet propagates back through the network, each node that receives the update sets its height to a value greater than the height of the neighbor from which the update was received. This has the effect of creating a series of directed links from the original sender of the query to the node that initially generated the update. An example of this process is illustrated in Figure 21.

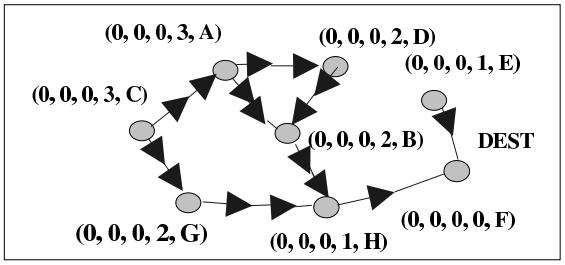
3.2.3.1.2 Maintaining Routes

Route maintenance is performed only for nodes that have a height other than null. When a node discovers that a route to a destination is no longer valid, it adjusts its height so that it is a local maximum with respect to its neighbors and transmits an update packet. If the node has no neighbors of finite height with respect to this destination, then the node instead attempts to discover a new route as describe above. Figure 22 is summaries the decision tree for maintaining routes. Route maintenance processes are illustrated in Figure 23 and Figure 24.



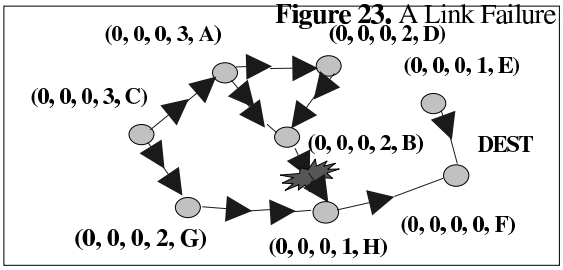


(a) Link D and E fails

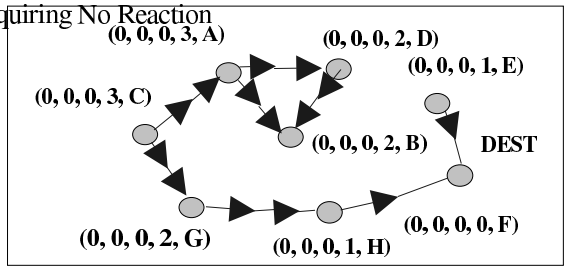


(b) No reaction necessary, all nodes still have downstream links

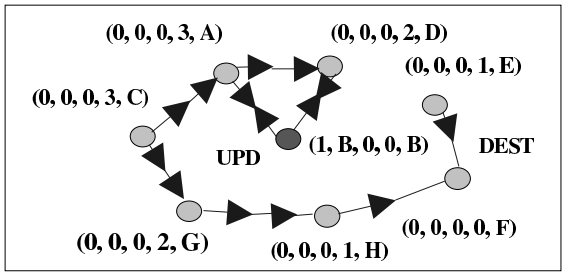
Figure 23. A Link Failure Requiring No Reaction



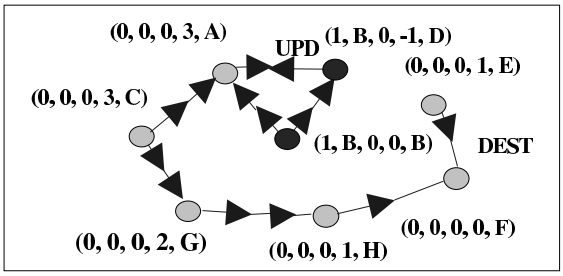
(a) Link B and H fails



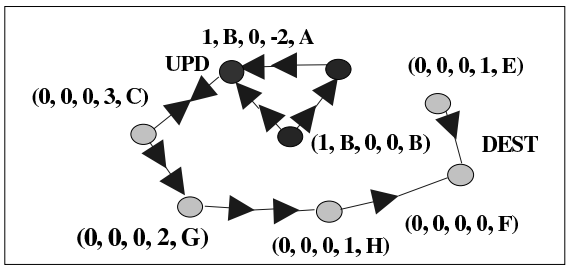
(b) B has no downstream links, so B initiates a reaction



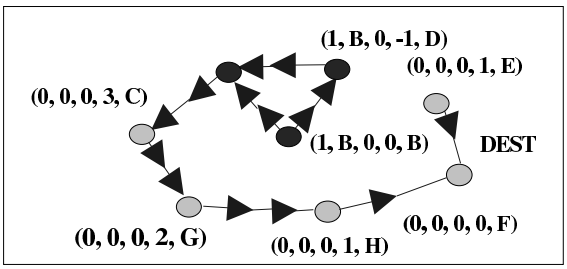
(c) Node B defines new reference level.
Since this height is the maximum height of the neighbors, all the neighboring links are revised.



(d) Node D propagates reference level
Since A's reference level is older, the ht. of D is less than B but greater than A. When D updates and reaches A, the link direction gets revised.



(e) Node A propagate reference level
A has no downstream links, it also sets its reference level to the current newer reference level and choose a offset of -2 (as it should be below D). This leads to link reversal between A and C.



(f) Failure reaction complete.
Since Node C still has one downstream link, it does not propagate the reaction

3.2.3.1.3 Erasing Routes

When a node detects a network partition, where a part of the network is physically separated from the destination, the node generates a *clear* packet that resets the routing state and removes invalid routes from the network. The process is shown in the Figure 25.

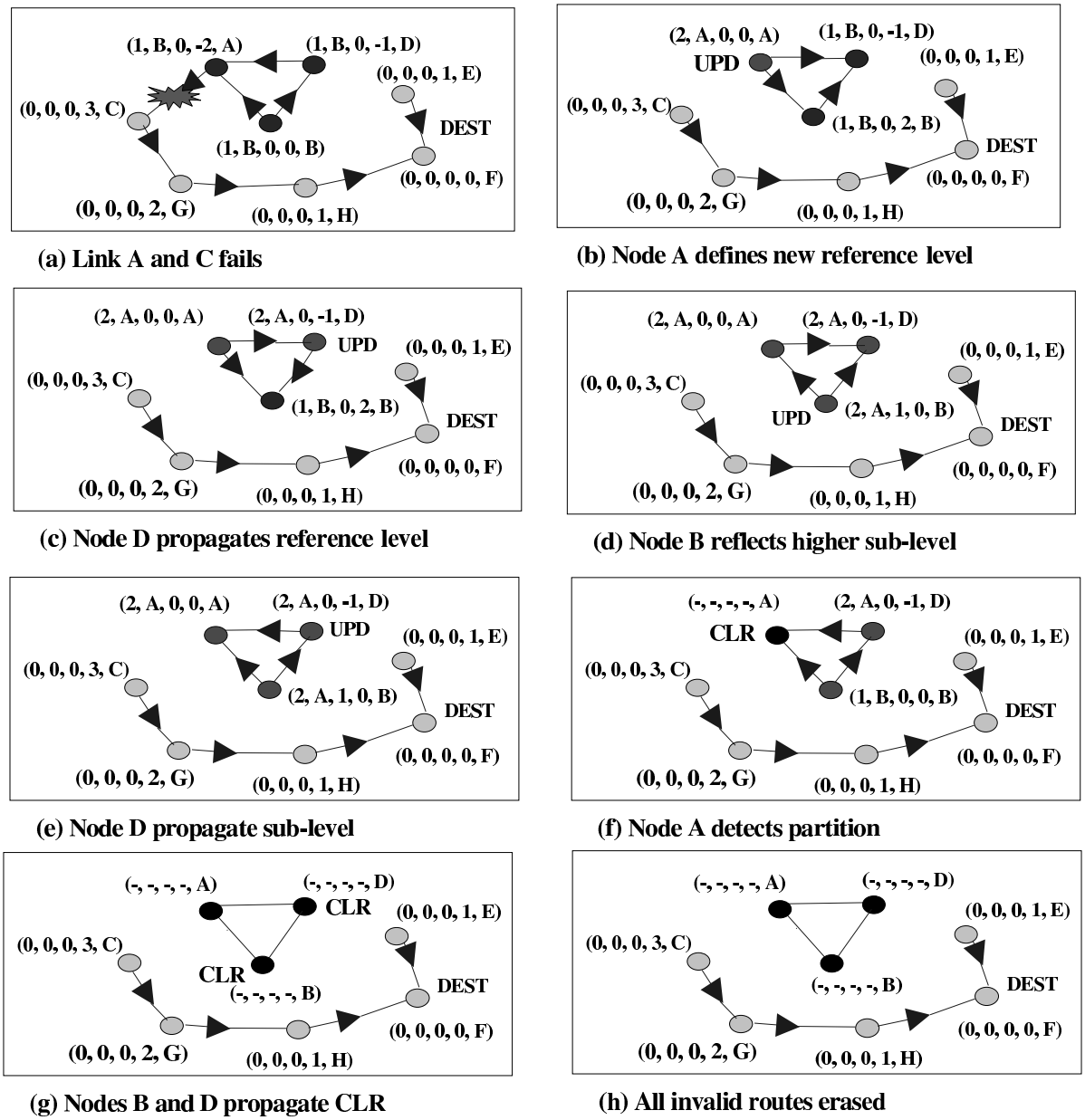


Figure 25. Erasing Process Following a Network Partition

3.2.3.1.4 Optimizing Routes

Proactive periodic route optimization is also possible which can be activated on a per-destination basis, under the control of the destination itself. Such functionality might be used when many (or all) nodes frequently require routes to a given destination, perhaps to a gateway router or a Mobile IP foreign agent [4].

Optimized TORA requires the definition of a new packet type, the optimized (OPT) packet. OPT packets for a given destination are generated only by the destination itself. Each node maintains an indication of the mode of operation for routing to the destination. The optimization Mode flag OM_i may be either OFF, PARTIAL or FULL-determining the behavior of periodic optimizations (if any) for the destination. Each node also maintains an optimization sequence number OSi, which indicates the most recently seen OPT packet received from the destination. For the optimization mechanism, the destination also maintains an optimization period, which determines the frequency with which periodic optimizations will occur.

The DEST controls the setting of the aforementioned parameters, which are disseminated to the nodes in OPT packets. The OPT packet contains several fields: the DEST identifier, the optimization mode set by the destination OM_{DEST} setting, the optimization sequence number set by the destination OS_{DEST}, and a delta value δ value set by the node that broadcast the packets.

If optimization is turned ON (i.e. OM_{DEST} = PARTIAL or FULL) for a DEST, the DEST node periodically generates OPT packets with successively higher sequence numbers. The extend of their dissemination is determined by the OM_{DEST} setting and height in the network. When a node i receives an OPT packet from some neighbor j, it checks to see if the sequence number in the packet OS_{DEST} is higher than the one it has previously seen, OS_i. If not, it discards the packet; if so, it updates its OS_i and OM_i values stored for the DEST.

In our above description we assumed that the nodes participating in the computation have synchronized clocks. If the nodes do not have synchronized clocks, the distributed computation can be shown to enter an erroneous condition in certain cases resulting in the formation of invalid routes to the destination.

TORA sits as a routing layer above another, network level protocol called the Internet MANET Encapsulation Protocol (IMEP), which is required to provide reliable, in-order delivery of all routing control messages from a node to each of its neighbors, plus notification to the routing protocol whenever a link to one of its neighbors is created or broken. IMEP is designed to support the operation of many routing algorithms, network control protocols or other upper layer protocols intended for use in mobile ad hoc networks. The protocol incorporates mechanisms for supporting link status and neighbor connectivity sensing, control packet aggregation and encapsulation, one-hop neighbor broadcast reliability, multipoint relaying, network-layer address resolution.

3.2.3.2 Complexities

The time complexity and communication complexities of TORA same as AODV and DSR.

3.2.3.3 Strengths and Weaknesses

- **Strengths:** TORA provides loop free paths at all instants. It provides multiple routes so that if one path is not available, another is readily available. It established routes quickly so that they may be used before the topology changes. Route reconstruction is not necessary until all known routes to a destination are deemed invalid, and thus conserves available bandwidth and increases adaptability. It is also able to detect network partitions very quickly.
- **Weaknesses:** TORA's reliance on synchronized clocks, while a novel idea, inherently limits its applicability. If a node does not have GPS or some other external time source, it cannot use the algorithm. If an external time source fails, the algorithm will cease to operate. Additionally, route

discovering in TORA may not occur as quickly as in the other algorithms due to the potential for oscillations during this period. This can lead to potentially lengthy delays while waiting for the new routes to be determined. Furthermore, it has very high overhead because of the use of IMEP.

3.2.4 Associativity-Based Routing (ABR)

The novel protocol is known as Associativity-Based Routing (ABR) is developed at Cambridge University. Associativity-Based Routing (ABR) [10, 11] is a protocol that is designed for an ad hoc mobile network environment. ABR is a compromise between broadcast and point-to-point routing and it uses the connection-oriented packet forwarding approach. Routes are established based on demand. It is free from loops, deadlock, and packet duplicates. The uniqueness of this scheme is the route selection criteria. By exploiting the spatial and temporal relationship of mobile hosts, ABR introduces the following new routing metrics: Longevity of a route based on associativity, Route relaying load of intermediate nodes supporting existing routes, and Link capacities of the selected route.

These metrics are known as the degree of association stability. In ABR, a route is selected based on associativity states of nodes. The routes thus selected are liked to be long-lived. All nodes generate periodic beacons to signify their existence. When a neighbor node receives a beacon, it updates its associativity tables. For every beacon received, nodes increment their associativity tick with respect to the node from which it received the beacon. Association stability means connection stability of one node with respect to another node over time and space. A high value of associativity tick with respect to a node indicates a low state of node mobility, while a low value of associativity tick may indicate a high state of

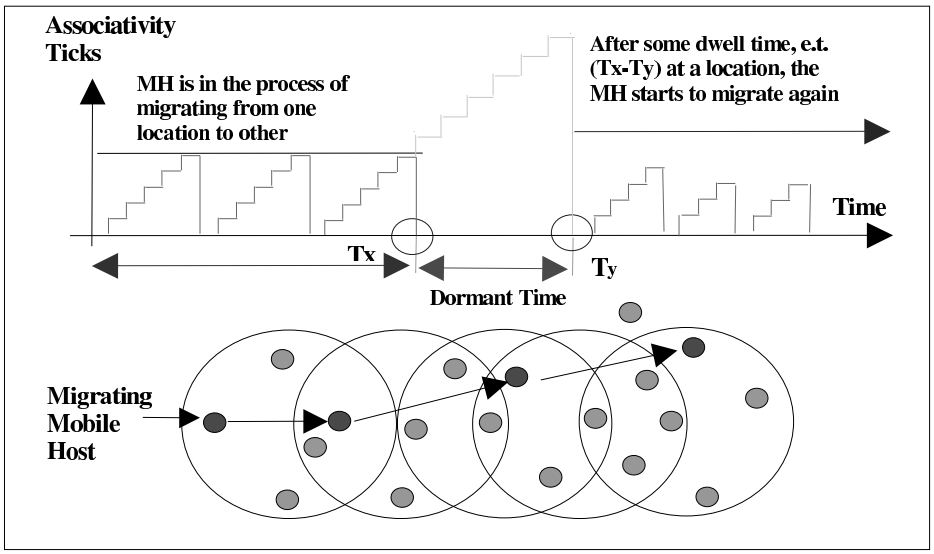


Figure 26. Time and spatial Representation of Associativity of a MH with its neighbors

node mobility. Associativity ticks are reset when the neighbors of a node or the node itself move out of proximity. The fundamental objective of ABR is to find longer-lived routes for ad hoc mobile networks. The process is illustrated in the above Figure 26.

3.2.4.1 Protocol Description

The ABR protocol consists of three phases, namely: Route Discovery Phase, Route Reconstruction (RRC) Phase, and Route Deletion Phase. Initially, when a source node desires a route, the route discovery phase is invoked. Then, if any of the three kinds of nodes (SRC, source node; DEST, destination node; IN, intermediate node) move, or if subnet bridging of a mobile host's migration occurs, reconstruction has to start. Finally, when the SRC no longer desires the route, it initiates the route deletion phase.

3.2.4.1.1 Route Discovery phase

The route discovery process consists of a Broadcast Query (BQ) and BQ- REPLY cycle. When a source demands a route to the destination, it broadcast a BQ message with unique sequence number, which is propagated throughout the ad-hoc mobile network in search of nodes that have a route to the destination. Any Intermediate Node (IN) that receives the BQ packet checks if the message has already been processed by looking up the seen table. If the BQ packet has not been seen before, it appends the

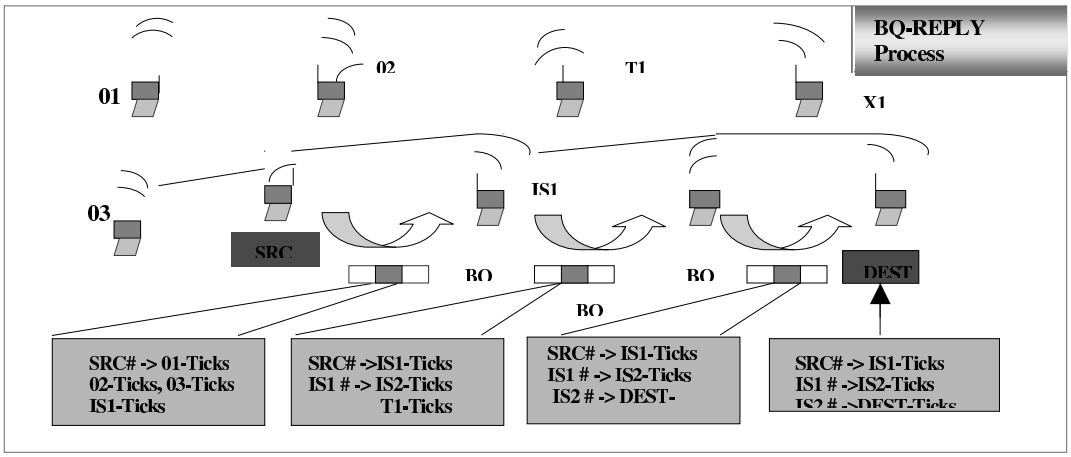


Figure 27. Updating Associativity Metric During BQ Packet

following to the BQ packet: (a) its identifier, (b) associativity ticks with its neighbors, (c) route relaying load, (d) link propagation delay, and (e) hop count information. The IN then broadcasts the packet to its neighbors.

When the destination node receives BQ packets, it knows all the possible routes and their qualities. The destination node then selects the best route based on longevity and other qualities (route load, minimum hop, etc.) and sends a BQ-REPLY control packet (which contains a list of INs' addresses IDs and a summary of selected route QoS) back to the source node via the selected route. Nodes propagating the REPLY mark their routes as valid. All other routes remain inactive, and the possibility of duplicate packets arriving at the destination is avoided.

Case When Source Never Receives REPLY: There may be some rare instances when the source never receives a destination's REPLY because of some unexpected 'not -yet -selected' INs' movement. In such circumstances, the source will eventually BQ-TIMEOUT and send another BQ query. Since the downstream neighbour of the migrating IN realizes the associativity change, it will send a RN (Route Notification) packet in the downstream direction, deleting all the downstream nodes' invalid routing table entries. Another situation occurs when a selected IN moves while the REPLY propagation is still in progress. The upstream neighbour of the migrating node will perform a LQ [H] (localised query) process to discover a new partial route, while the downstream neighbour sends a RN [1] packet towards the destination, thereby erasing all invalid downstream nodes' routing entries. Hence, while the RRC is in progress, the reply packet continues to propagate towards the source. Figure 28a and 28b illustrate these two scenarios.

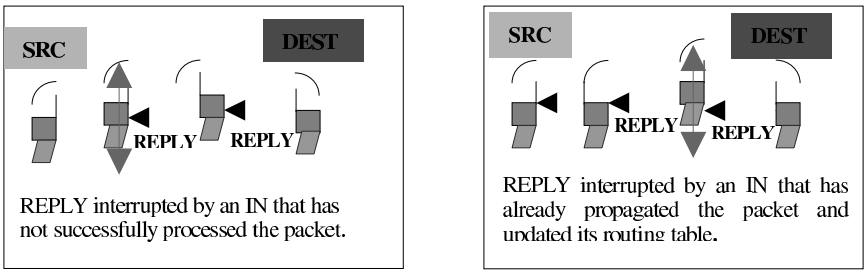


Figure 28. REPLY Interruption Caused By Unexpected INs'

3.2.4.1.2 Route Reconstruction Phase (RRC)

In the ABR protocol, the selected route is more likely to be long-lived due to the property of associativity. However, if unexpected moves occur, the RRC procedures will attempt to quickly locate an alternative valid route without resort to broadcast query unless necessary. The RRC phase consists of partial route discovery, invalid route erasure, valid route updates, and new route discovery, depending on which node(s) along the route move. Source node movement results in a new BQ-REPLY process, as shown in Fig. 28a as the routing protocol is source-initiated. The route notification RN[1] message is used to erase the route entries associated with downstream nodes.

When the destination moves, the destination's immediate upstream node erases its route. A localized query (LQ[H]) process, where H refers to the hop count from the upstream node to the destination (Fig. 28b), is initiated to determine if the node is still reachable. If the destination receives the LQ packet, it selects the best partial route and REPLYs; otherwise, the initiating node times out and backtracks to the next upstream node. An RN[0] message is sent to the next upstream node to erase the invalid route and inform this node that it should invoke the LQ [H] process. If this process results in backtracking more than halfway to the source, the LQ process is discontinued and the source initiates a new BQ process.

When any IN moves, the immediate upstream node will invoke an LQ process to quickly locate an alternate partial route. However, the immediate downstream node will immediately send a *route erase* message towards the DEST. In this manner, invalidated route entries are deleted. Again, multiple partial routes can exists, and the destination node will select the best possible route. If no partial routes to the DEST exist, the next upstream node invokes another LQ process. This backtracking proceeds until a partial route is found or until the number of backtrack steps exceeds half the route length. If all possible localized queries are unsuccessful, the source will timeout and may invoke a BQ instead.

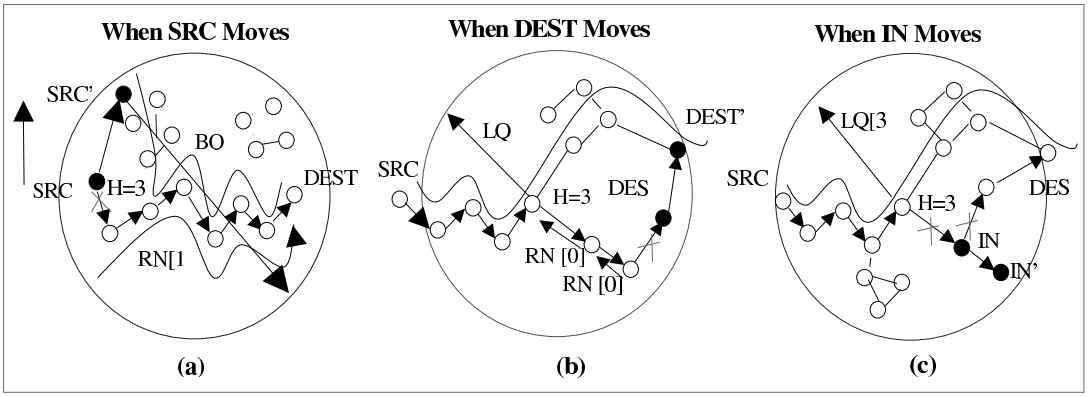


Figure 29. Route maintenance when SRC, DEST and IN moves.

Race condition exists because of multiple invocations of RRC processes as a result of concurrent movements by the SRC, the DEST and the INs. However, the ABR protocol is able to resolve the conflicts of multiple RRCs by ensuring that only one ultimately succeeds. Each LQ process is tagged with a sequence number so that earlier LQ processes will terminate when a new one is invoked. In the same vein, if nodes processing LQs hear a new BQ for the same connection, the localized query process is aborted.

3.2.4.1.3 Route Deletion Phase

When a discovered route is no longer needed, the source node initiates a route delete (RD) broadcast. All nodes along the route delete the route entry from their routing tables. The RD message is propagated by a

full broadcast, as opposed to a directed broadcast, because the source node may not be aware of any route node changes that occurred during RRCs.

3.2.4.2 Acknowledgement and Retransmission

- **Data Flow Acknowledgement:** To utilize the channel efficiently, ABR uses a simple short packet header. Each data packet header contains only the neighboring node information rather than all the nodes in the routes.

  Flow control is achieved by monitoring passive acknowledgements. When a node receives a packet and performs relaying via a radio transmission to its neighbors, its previous neighbor that has sent it the packet will have heard the transmission and hence this is indirectly used as an acknowledgement to the packet sent. Active acknowledgement is used by the destination node (since it has no more neighbors to relay the packet to) where an explicit message is sent to the upstream node.

- **Packet Retransmission:** If a node does not receive a passive acknowledgement within the timeout period after forwarding a packet, it retransmits the data packet for an appropriate number of times. If an acknowledgement is not received after a few attempts, a mobile host is considered to have moved

out of radio range or has powered down and a RRC phase is therefore invoked, as illustrated in Figure 30

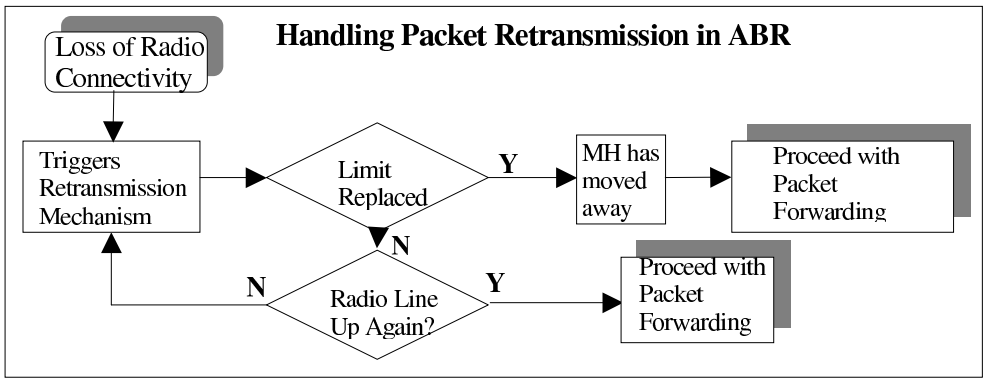


Figure 30. Handling Packet Retransmission in ABR

3.2.4.3 Strengths and Weakness

Strengths: The major advantage of ABR is that routes selected tend to be long-lived since nodes, which have been stationary for some time, are less likely to move. This results in fewer route reconstructions, thereby reducing the communication overhead and yielding higher throughput. In addition, like DSR, ABR also does not attempt to consistently maintain routing information in every node reducing the communication overhead further. Another advantage of ABR is that, it is guaranteed to be free of packet duplicates.

Weakness: Since when a route is broken due to intermediate link failure, the back tracking done in partial route discovery could entail a high latency, though at times it could provide quicker route maintenance. So, there is a trade-off involved. In case of partitioned networks, it will take a lot of time for a source to

ultimately detect it and will also result in waste of resources. Another possible drawback of ABR is that ABR may have scalability problem due to the limited bandwidth. In addition, Periodic beaconing requirement may result in additional power consumption.

3.2.4.4 ABR Protocol Summary

Figure 31 summarizes the procedures of the routing protocol under different node's associativity states. When this property is violated, the protocol will invoke a LQ or BQ process to quickly locate alternate route.

<table>
<tr><td>Associativity Valid</td><td colspan="6">Associativity Violated</td></tr>
<tr><td rowspan="2">No Route Re - Constructions Are Needed</td><td colspan="2">INs & DEST Moves</td><td>SRC Moves</td><td colspan="2">Subnet Bridging MH Moves</td><td>Concurrent Moves</td></tr>
<tr><td>Normal Case</td><td>Worst Case</td><td rowspan="2">BQ-REPLY Cycle Success</td><td>Route Within Subnet</td><td>Route Spans Across Subnet</td><td rowspan="2">Ultimately Only One Route Reconstruction Cycle is Valid</td></tr>
<tr><td>LQ-REPLY Cycle Success</td><td>BQ-REPLY Cycle Success</td><td>No Route Re-constructions Are Needed</td><td>Networks is Partitioned BQ-REPLY Cycle will retry before aborting</td></tr>
</table>

Figure 31. Summary of Route Reconstructions under varying node's migration

3.2.4.5 Complexity Comparison

To further illustrate the simplicity and efficiency of the ABR protocol, comparisons with some existing routing algorithms in terms of time and communication complexities are made, as shown in Table 10.

Types of Routing Protocol	*Time Complexity*	*Communication*
ILS	O (d)	O (2E)
MS	O (d^2)	O (N^2)
DBF (link failure)	O (N)	O (N^2)
NP (QRY-RPY, initialization)	O (2d)	O (2N)
NP (QRY-RPY, postfailure)	O (2l)	O (2x)
NP (FQ-RPY, postfailure)	O (2l)	O (2x)
ABR (QRY-RPY, initialization)	*O (d+z)*	*O (N+y)*
ABR (QRY-RPY, postfailure)	*O (l+z)*	*O (x+y)*
Flooding (no control packets)	0	0

Table 10. Complexity Comparison of ABR with Some Existing Routing Protocols

Abbreviations: (refer to tables 11 and 12 for N, d, x, l, y, z)
E = Number of links in the network
ILS = Idealized Link-State protocol
MS = Merlin-Segall
DBF = Distributed Bellman-Ford algorithm
NP = Routing protocol proposed in (44).

3.2.5 Signal Stability Routing

Signal Stability-Based Adaptive Routing protocol (SSR) presented in [48] is an on-demand routing protocol that selects longer-lived routes based on the signal strength between nodes and a node's location stability. This route selection criterion has the effect of choosing routes that have "stronger" connectivity and presumably are more stable and require less route maintenance.

3.2.5.1 Protocol Overview

A source initiates a route discovery request when it has a packet to send to some destination, which is not in the routing table. The route-search is broadcast to all neighboring hosts. These hosts propagate the broadcasts if (a) it is received over a strong channel and (b) the request has not been propagated previously (to avoid looping). The route-search packet stores the address of each intermediate host in the route taken. The destination chooses the route recorded in the first arriving request, since this route is probably shorter and less congested than routes for slower arriving request. The destination returns the route-reply along the selected route, and each intermediate node includes the new next-hop, destination pairs in its routing table.

Functionally, the Signal Stability based Adaptive Routing (SSR) protocol consists of two cooperative protocols: the Dynamic Routing Protocol (DRP) and the Static Routing Protocol (SRP). The DRP maintains the routing table and SRP performs the actual routing table lookup to forward a packet onto the next-hop.

3.2.5.2 Protocol Modules

The Static Routing Protocol (SRP) and the Dynamic Routing Protocol (DRP) work together to route packets in the ad-hoc network. The DRP maintains the Signal Stability Table (SST) and Routing Table (RT). The SST stores the signal strength of neighboring nodes obtained by periodic beacons from the link layer of each neighboring node. Signal strength is either recorded as a strong or weak channel. All transmissions are received by DRP and processed. After updating the appropriate table entries, the DRP passes the packet to the SRP.

The SRP passes the packet up the stack if it is the intended receiver. If not, it looks up the destination in the RT and forwards the packet. If there is no entry for the destination in the RT, it initiates a route-search process to find a route. Route-request packets are forwarded to the next hop only if they are received over strong channels and have not been previously processed (to avoid looping). The destination chooses the first arriving route-search packet to send back as it is highly likely that the packet arrived over the shortest and/or least congested path. The DRP reverses the selected route and sends a route-reply message back to the initiator of the route-request. The DRP of the nodes along the path update their RTs accordingly.

A route may become unavailable due to migration of the hosts along the route. When a host moves out of range of its neighbors or shuts down, the neighbors will recognize that the host is unreachable since they no longer receive beacons from that host. The DRP will modify the SST and RT to reflect the changes. Any routes that have this unreachable host as the next-hop will become invalid. When the host receives a packet to forward along an invalid route, SRP will determine the lack of a route and will notify the source via an error message. The source SRP will initiate a new route discovery to find an available route, and it will send a message to erase the invalid route. The process is shown in Figure 32 (a) and (b) below.

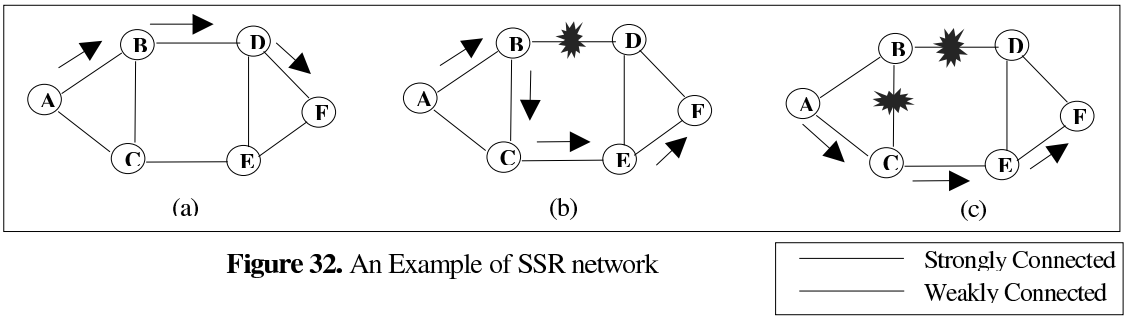


Figure 32. An Example of SSR network

3.2.5.3 Route Maintenance

Route maintenance is triggered when a host has data to send over a fail link. When a link failure is detected within the network, the intermediate nodes send an error message to the source indicating which

channel has failed. The source then sends an erase message to notify all nodes of the broken link and initiates a new route-search process to find a new path to the destination. Error and erase messages are unicast with best-effort delivery and are dropped if a host is unable to forward the packet.

In case of multiple failures, some routing messages may not reach their destinations. This may result in the existence of stale routes, but it will not cause any routing errors or loops. The stale routes will be discovered and erased by the next data packet that tries to use the invalid route.

If multiple link failures occur nearly simultaneously, erase and error messages may not reach their destinations. In case of error packet cannot be delivered to the source, it must be due to another link failure, which will also trigger an error packet closer to the source. This second error packet will inform the source of the second failure so that the source can take appropriate action.
The process is illustrated in Figure 33 below.

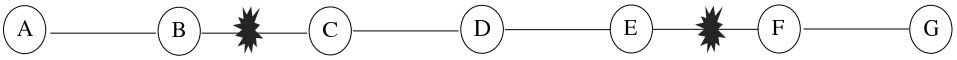


Figure 33. Failure of route erase packet

3.2.5.4 Optimization

There are a few optimizations in SSR, which can increase performance and reduce route discovery latency.

- **Route quality option:** Route-search packets arriving at the destination have necessarily arrived on the path of strongest signal stability because the packets arriving over a weak channel are dropped at intermediate nodes. If the source times out before receiving a reply then it changes the PREF field in the header to indicate that weak channels are acceptable, since these may be the only links over which the packet can be propagated (refer to Figure 1c).
- **Route Discovery Latency:** This optimization decreases route discovery latency and route-search propagation by allowing intermediate hosts to participate in route discovery. If an intermediate host receiving a route-search already has a route to the destination, it may send a route-reply immediately back to the source, to decrease the latency, in addition to forwarding the route-search to the destination. If this route is non-optimal or stale, it will be overwritten later by route-reply from the destination.

3.2.5.5 Complexities

The time complexity and communication complexities of SSR same as ABR.

3.2.5.6 Strengths and Weaknesses

Strengths: The major advantage of SSR is that routes selected tend to be long-lived and this in turn reduces the number of route reconstructions required. Furthermore, SSR results in low packet loss since strong links are less vulnerable to interference than weak links.

Weaknesses: A possible drawback of SSR is that each time a route failure occurs the route discovery protocol which consists of a query-reply process needs to be invoked. SSR also results in routes with slightly higher hop counts than optimal routing since SSR prefers routes with strong links, which are likely to be between two hosts close to each other. The simulation result shown in [48] indicated that the hop count is within a factor of 1.1 and 1.5 of the optimal hop count.

3.3 Hybrid Routing Protocol

3.3.1 Zone Routing Protocol (ZRP)

Zone Routing Protocol (ZRP) [66] is a hybrid of a reactive and a proactive routing protocol. It divides the network into several routing zones and specifies two totally detached protocols that operate inside and between the routing zones. A zone is a local region defined by a single parameter called the *zone radius,* which is measured in hops. Nodes proactively maintain routing information for nodes within their zones and reactively discover routes for nodes outside their zones. The two routing mechanisms are referred to as the Intrazone Routing Protocol (IARP) and Interzone Routing Protocol (IERP) respectively.

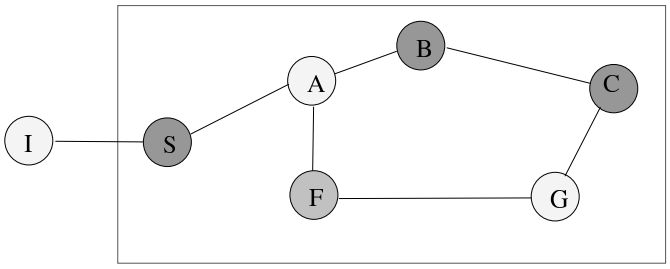


Figure 34. Zone of node F of radius 2

Figure 34 illustrates the zone of node F where the zone radius is two. The only node outside of F's zone is I. Although one path to it is of length three, C is in the zone because there is a two-hop path from F.

Peripheral nodes refer to nodes whose minimum distance from the node in question is the zone radius. In Figure 34; S, B and C are F's peripheral nodes. Nodes use their peripheral nodes when issuing a network-wide search for a route. By targeting queries to peripheral nodes, route requests tend to diverge away from the source, thereby reducing redundancy.

The strategy of using peripheral nodes for reactive route discoveries is called *bordercasting*. As the zone radius approaches the radius of the network, ZRP behaves like a proactive routing protocol. At the other extreme, where the zone radius is one, ZRP behaves like a reactive protocol. When the zone radius lies in between these two extremes, ZRP leverages the benefits of each to provide a robust, efficient, hybrid protocol.

3.3.1.1 Intrazone Routing Protocol (IARP)

IARP is the proactive component of ZRP. It operates inside the routing zone and learns the minimum distance and routes to all the nodes within the zone. The protocol is not defined and can include any number of proactive protocols, such as Distance Vector or link-state routing. Different zones may operate with different intrazone protocols as long as the protocols are restricted to those zones. A change in topology means that update information only propagates within the affected routing zones as opposed to affecting the entire network.

3.3.1.2 Interzone Routing Protocol (IERP)

IERP is the reactive part and is used for finding routes between different routing zones. IERP can be implemented as a reactive, distance vector algorithm, where nodes cache route information and packet routing is performed on a hop-by-hop basis. Alternatively, it can require no caching, in which case it relies on source routing. This is useful if the destination node does not lie within the routing zone.

When a node needs to send packets to a destination, it first checks to see if the destination is in the same zone. If so, the path to the destination is known (through IARP) and delivers them accordingly. If the destination is not within the source's routing zone, the source bordercasts a route query to all of its peripheral nodes, which in turn forwards the request if the destination node is not found within their routing zone. This procedure is repeated until the requested node is found and a route reply is sent back to

the source indicating the route. IERP uses a Bordercast Resolution Protocol (BRP) [66] that is included in ZRP. BRP provides bordercasting services, which do not exist in IP. Bordercasting is the process of sending IP datagrams from one node to all its peripheral nodes. BRP keeps track of the peripheral nodes and resolves a bordercast address to the individual IP addresses of the peripheral nodes. The message that was bordercasted is then encapsulated into a BRP packet and sent to each peripheral node.

An example of this route discovery procedure is shown in Figure 35. Nodes A, F, B, C, G and H all are in zone F. Even though node B also has a distance of 3 hops from node F, it is included in the zone since the shortest distance is only 2 hops. In Figure 35 nodes S, B, C and H are border nodes to F. Node S wants to send a packet to node D. Since node D is not in the routing zone of S, a route request is sent to the border nodes B and F. Each border node checks to see if D is in their routing zone. Neither B nor F finds the requested node in their routing zone; thus the request is forwarded to the respectively border nodes. F sends the request to S, B, C and H while B sends the request to S, F, E and G. Now the requested node D is found within the routing zone of both C and E thus a reply is generated and sent back towards the source node S.

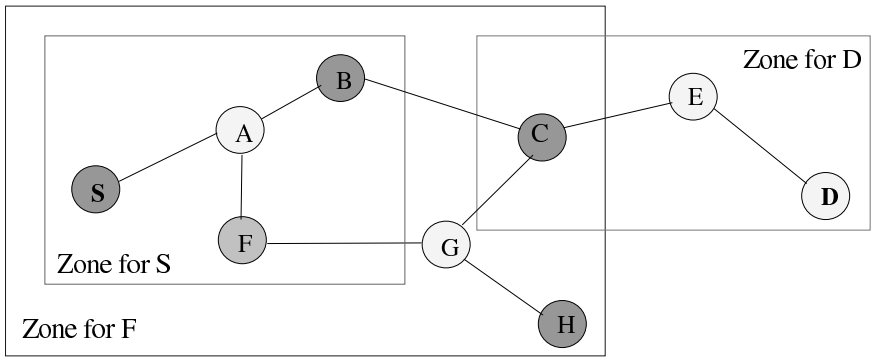


Figure 35. Network using ZRP. The three different colors shows three different zones F, S and D respectively

IERP must maintain route information during the propagation of route requests in order that the discovering node may transmit the route reply message back to the source node. The route reply message must also contain the newly discovered route from the source to the destination. Each node may place its IP address into the header of the route request packet as it propagates towards the destination. The discovering node may then reverse the route to send the route reply packet to the original source. If nodes do not cache any route information, strict source routing must be used. If some route information is cached, then loose source routing may be used. If all nodes cache route information, then next-hop routing may be used.

To reduce redundant route request propagation through zones that have already been queried, ZRP may incorporate a query detection and early termination mechanism. In such a scheme, nodes keep track of the queries they hear on the network. They detect the queries either by relaying them to peripheral nodes or by operating in promiscuous mode. The query source IP address and unique identifier (i.e. sequence number) are recorded in a Detected Queries table. If a redundant query enters a region, then the detecting node may drop the packet and discontinue its propagation in that area. ZRP offers two distinct methods of query detection for redundant queries and to reduce overhead. Query Detection 1 (QD1) allows the intermediate nodes to detect a redundant query and terminate the thread. Query Detection 2 (QD2) allows all nodes to detect a redundant query and terminate the request.

Figure 36 illustrates both levels of advanced query detection. In this example, node S broadcasts to two peripheral nodes, B and D. The intermediate nodes A and C can detect passing route request packets and record that S's routing zone has been queried. In single channel networks, node E may also be able to receive A's transmission and record the query information as well.

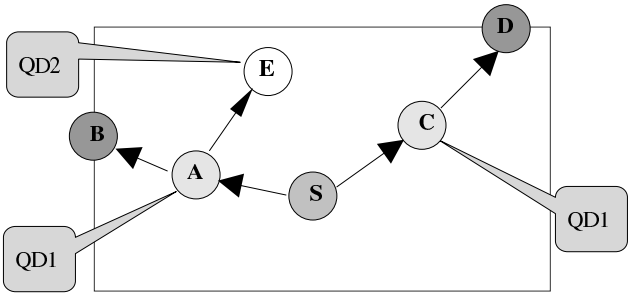


Figure 36. Advanced Query Detection (QD1/QD2)

A node will not relay a query packet to a targeted bordercast recipient either if that recipient lies inside the routing zone of a previously bordercast node or if this node has already relayed the query to this recipient. This scheme, which we refer to as early termination (ET), relies on query detection to identify which local nodes have already bordercast the query. To identify the nodes that lie within the routing zones of these bordercast nodes, the topology of an extended zone of radius 2ρ-1 hops (where ρ is the radius of the "basic" routing zone) must be maintained. Conveniently, IARP already maintains this extended information in support of multicast based border casting.

Figure 37 illustrates the operation of ET. Node B first detects (and relays) a route request packet broadcast by node T. Later, B receives a route request packet to be relayed to the broadcast recipient node C. B recognizes that node C belongs to the previously queried routing zone of node T and therefore withholds transmission.

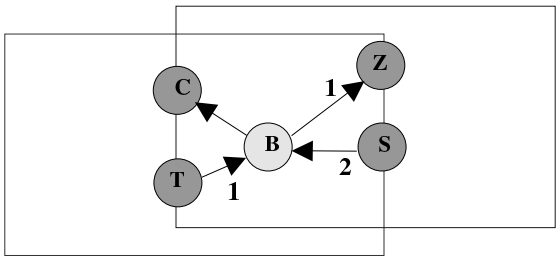


Figure 37. Early termination

IERP also provides route maintenance. Route failures can be detected and reported proactively by IARP when a node leaves a zone. They can also be reported by IP when the next-hop for a datagram is determined to be unreachable. Nodes detecting a route failure may choose to notify the source and/or attempt to repair the route.

3.3.1.3 Optimization

A mathematical expression for the optimum zone radius for optimum performance has not yet been determined [64]. Even with perfect knowledge of all network parameters, computation of an optimal routing zone radius is not a straightforward mechanism. Haas [64] recommends that further research could focus on complete derivation of the ZRP traffic function. As depicted in Figure 38, a simple approach is to adjust the zone routing radius until the setting for minimum ZRP overhead traffic is achieved. In the interim, two other schemes have been suggested for optimum zone routing radius selection: min-searching and traffic adaptive method [64]. Min-searching assumes that the node behavior will not change quickly over a period of time and an accurate assessment of ZRP traffic can be obtained. As shown in Figure 38, if IERP traffic is decreasing and the amount of IARP traffic is increasing, there is an "undershoot" of the optimum zone radius. Likewise, if IERP traffic is increasing and IARP traffic is also increasing would indicate an "overshoot" of the optimum zone radius. IARP traffic volume depends on routing protocol. If different nodes manage these zones with different protocols, different optimal radii will result.

The traffic adaptive method only relies on current estimates. In this case, Haas [64] has shown that the amount of ZRP traffic generated is significantly higher when the ZRP traffic is dominated by the reactive

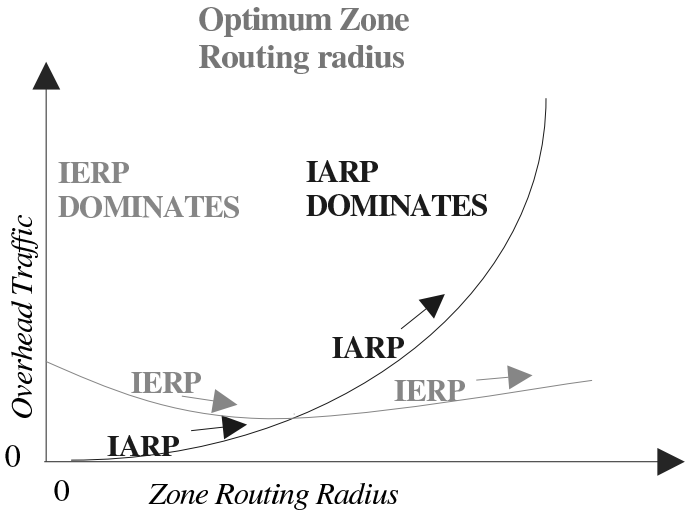


Figure 38. ZRP Zone Routing Radius Optimization

IERP query traffic. As depicted in Figure 38, the optimum region resides between these two regions. In other words, the ratio between IERP to IARP (IERP/IARP) should be as close to one as possible for optimization. The general rule-of-thumb is that a sparse network favors a large routing zone and a dense network favors a small routing zone.

3.3.1.4 Strengths and Weaknesses

Strengths: Since this is a hybrid between proactive and reactive schemes, this protocol use advantages from both. Routes can be found very fast within the routing zone, while routes outside the zone can be found by efficiently querying selected nodes in the network.

Weaknesses: The proactive intrazone routing protocol is not specified. The use of different intrazone routing protocols would mean that the nodes would have to support several different routing protocols. This is not a good idea when dealing with thin clients. It is better to use the same intrazone routing protocol in the entire network. ZRP also limits propagation of information about topological changes to the neighborhood of the change only (as opposed to a fully proactive scheme, which would basically flood the entire network when a change in topology occurred). However, a change in topology can affect several routing zones. In addition, since each node has higher level topological information, memory requirement is greater.

ZRP is very interesting protocol that divide the network into several zones. This approach is probably a very good solution for large networks. Within the zones it has a more proactive scheme and between the zones it has a reactive scheme that has many similarities with the operation of AODV and DSR. ZRP has for instance a route discovery phase that sends request through the network.

4.0 Unicast Comparison

So far, the protocols have been analyzed theoretically. Sections 4.1 and 4.2 summarized and compared the results from these theoretical/quantitative analyses for table-driven and on demand protocols respectively. In Table 10 and 11, time complexity is defined as the number of steps needed to perform a protocol operation, and communication complexity is the number of messages needed to perform a protocol operation [19, 11, 44].

4.1 Table-Driven protocols

This section compares table-driven protocols based on Table 11. DSDV routing is essentially a modification of the basic Bellman-Ford routing algorithm. It guarantees loop-free routes and a simple route update. DSDV allows the protocol to converge in reasonable time. However, mobility cannot be too high. While only providing one path to any given destination, DSDV selects the shortest path based on the number of hops to the destination. It has two types of update messages, a large one and significantly smaller one. The smaller one can be used for incremental updates instead of rebroadcasting the entire network structure whenever the topology changes. However, DSDV relies on periodic update transmissions, producing an overhead which grows as $O(N^2)$. In CGSR, DSDV is used as the underlying routing protocol. Routing in CGSR occurs over cluster heads and gateways. A cluster head table is

necessary in addition to the routing table. One advantage of CGSR is that several heuristic methods can be employed to improve the protocol's performance. These methods include priority token scheduling, gateway code scheduling, and path reservation.

GSR is functionally similar to link state routing in that it maintains a topology map at each node. Whenever a node detects a topology change, LS packets are generated and flooded. In GSR, LS packets are not flooded. The GSR periodic table exchange resembles the vector exchange in DSDV, where the distances are updated according to the timestamp or sequence number assigned by the node originating the update. The drawbacks of GSR are the large size update messages that consume a considerable amount of bandwidth.

FSR achieves control traffic reduction by selectively adjusting routing update frequencies, while HSR reduces the size of update messages by using a hierarchical addressing approach. FSR maintains a flat addressing scheme and topology map. This makes it easy to locate destinations, but limits scalability because of routing table storage and processing overhead. HSR, in contrast, resolves the routing table scalability problem by using the hierarchical approach. However it must face the difficult problem of "finding" the destination. This problem can be resolve with a home agent technique, which extends the mobile IP concept to the multihop mobile environment.

WRP requires each node to maintain four routing tables, which can lead to substantial memory requirements, especially when the number of nodes in the network is large. In addition, whenever there are no recent packet transmissions from a given node, WRP requires the use of hello packets within a specified time period to ensure connectivity. The hello packets consume bandwidth and disallow a node to enter sleep mode. However, although it belongs to the class of path-finding algorithms, WRP has an advantage over other path finding algorithms because it avoids the problem of creating temporary routing loops that these algorithms have through the verification of predecessor information, as described in the earlier section. As can be seen from the above Table 11, none of the protocols support power conservation, security and QoS.

It is now important to highlight the differences of table-driven protocols discussed so far. During link failures, WRP has lower time complexity than DSDV since it only informs neighboring nodes about link status changes. During link additions, hello messages are used as a presence indicator such that the routing table entry can be updated. Again, this only affects neighboring nodes. In CGSR, because routing performance is dependent on the status of specific nodes (cluster head, gateway, or normal nodes), time complexity of a link failure associated with a cluster head is higher than in DSDV, given the additional time needed to perform cluster head reselection. Similarly, this applies to the case of link additions associated with the cluster head. There is no gateway selection in CGSR since each node declares itself as a gateway node to its neighbors if it is responding to multiple radio codes. If a gateway node moves out of range, the routing protocol is responsible for routing the packet to another gateway.

Parameters	DSDV	CGSR	GSR	FSR	HSR	WRP
Time Complexity (link addition/failure)	O (d)	O (d)	O (N^2)	O (N^2)	O (N^2)	O (h)
Communica'n complexity (link addition/failure)	O (x=N)	O (x=N)	O (N^2)	O (N^2)	O (N^2)	O (x=N)
Routing Philosophy	Flat	Hierarchical	Flat	Flat	Hierarchical	Flat*
Loop-free	Yes	Yes	Yes	Yes	Yes	Yes, but not instantaneous
Multicast	No	No**	No	No	No	No
Number of required tables	Two	Two	Four	Three	Two	Four
Freq. of update transmissions	Periodically, as needed	Periodically	Periodically	Periodically	As needed	Periodically, as needed
Update transmitted to	Neighbors	Neighbors & CH	Neighbors	Neighbors (Partial)	Neighbors & CH	Neighbors
Utilizes seque. numbers	Yes	Yes	Yes	Yes	No	Yes
Utilizes hello messages	Yes	No	No	No	No	Yes
Critical nodes	No	Yes	No	No	Yes	No
Routing metric	Shortest path	Shortest path	Shortest path	Shortest path	HID	Shortest path
Link type	Symmetric	Symmetric	Symmetric	Symmetric	Symmetric	Symmetric
Power conservation	No	No	No	No	No	No
Network size	Small	Small-med.	Large	Large	Large	Small
Requires reliable or sequenced data	No	No	No	No	No	No
Security	No	No	No	No	No	No
Quality of Service	No	No	No	No	No	No

Abbreviations:
N=Number of nodes in the network
d=Network diameter
h=Height of routing tree
x=Number of nodes affected by a topological change
CH=Cluster head
** While WRP uses flat addressing, it can be used hierarchically*
*** A separate protocol runs on top of CGSR provides multicast capability*

Table 11. Comparisons of the characteristics of table-driven routing protocols.

In terms of communication complexity, since DSDV, CGSR, and WRP use distance vector shortest-path routing as the underlying routing protocol, they all have the same degree of complexity during link failures and additions. On the other hand GSR, FSR, and HSR all have same the but different complexity than DSDV, CGSR and WRP.

4.2 Source-Initiated On-Demand Routing Protocols

This section compares on-demand protocols. Table 12 presents a comparison of AODV, DSR, TORA, ABR and SSR. AODV shares certain salient characteristics with DSR. In particular, they both discover routes only when data packets lack a route to a destination. Route discovery in either protocol is based on a query and reply cycle, and route information is stored in all intermediate nodes along the route in the form of route table entries (AODV) or in route caches (DSR). However, there are several important differences in the dynamics of these two protocols, which may give rise to significant performance differentials. The most notable of these is that the overhead of DSR is potentially larger than that of

AODV since each DSR packet must carry full routing information, whereas in AODV packets only contain the destination address. Similarly, the route replies in DSR are larger because they contain the address of every node along the route, whereas in AODV route replies only carry the destination IP address and sequence number. Also, the memory overhead may be slightly greater in DSR because of the need to remember full routes, as opposed to only next hop information in AODV. A further advantage of AODV is its support for multicast [8]. None of the other unicast algorithms considered in this project currently incorporate pure multicast communication. On the down side, AODV requires symmetric links between nodes, and hence cannot utilize routes with asymmetric links. In this aspect DSR is superior, since it does not require the use of such links and can utilize asymmetric links when symmetric links are not available.

Parameters	AODV	DSR	TORA	ABR	SSR
Time Complexity (initialization)	O (2d)	O (2d)	O (2d)	O (d+ z)	O (d + z)
Time complexity (postfailure)	O (2d)	O (2d) or 0*	O (2d)	O (l+ z)	O (l+ z)
Communication complexity (initialization)	O (2N)	O (2N)	O (2N)	O (N+ y)	O (N+ y)
Communication complexity (postfailure)	O (2N)	O (2N)	O (2x)	O (x+ y)	O (x+ y)
Routing Philosophy	Flat	Flat	Flat	Flat	Flat
Loop-free	Yes	Yes	Yes	Yes	Yes
Multicast capability	Yes	No	No**	No	No
Beaconing requirements	No	No	No	Yes	Yes
Multiple route possibilities	No	Yes	Yes	No	No
Route Maintained in	Route table	Route table	Route table	Route table	Route table
Utilizes route cache/table expiration timers	Yes	No	No	No	No
Routing reconfiguration methodology	Erase route, notify source	Erase route, Notify source	Link reversal, route repair	Localized broadcast query	Erase route, notify source
Routing metric	Freshest and shortest path	Shortest path	Shortest path	Associativity, shortest path & others***	Associativity and stability
Assured link type	Symmetric	Asymmetric	Asymmetric	Symmetric	Symmetric
Power conservation	No	No	No	Yes	Not specified
Network size	Small-medium	Small-medium	Large	Small-medium	Small-med.
Required reliable or sequenced data	No	No	Yes	No	No
Security	No	No	No	No	No
Quality of Service	No[a]	No[a]	No	No	No

Abbreviations: (refer to table 11 for d, h, x, N)
l = Diameter of the affected network segment
z = Diameter of the directed path where the REPLY packet transmit
y = Total number of nodes forming the directed path where the REPLY packet transmit
a = supports certain quality of service parameters; in particular, maximum Delay and Minimum Bandwidth.
** Cache hit,*
***LNM which runs on top of TORA provides multicast capability*
****ABR also uses the route relaying load and cumulative forwarding delay as routing metrics.*

Table 12. Comparisons of the characteristics of source limited on-demand ad hoc routing protocols

By virtue of source routing, DSR has access to a significantly greater amount of routing information than AODV. In particular, it can learn routes to every node on the source route of that data packet. In the absence of source routing and promiscuous listening, AODV can gather only a very limited amount of routing information. In particular, route learning is limited only to the source of any routing packets being forwarded. This usually causes AODV to rely on a route discovery flood more often, which may carry significant routing overhead. DSR does not make use of periodic routing advertisements, thereby saving

bandwidth and reducing power consumption. Hence, the protocol does not incur any overhead when there are no changes in network topology. Additionally, to make use of route caching aggressively, DSR replies to all requests reaching a destination from a single request cycle. Thus, the source learns many alternate routes to the destination, which will be useful in the case that the primary (shortest) route fails. Having access to many alternate routes saves route discovery floods, which is often a performance bottleneck. However, there may be a possibility of a route reply flood. In AODV, on the other hand, the destination replies only once to the request arriving first and ignores the rest. The routing table maintains at most one entry per destination.

Furthermore, the current specification of DSR does not contain any explicit mechanism to expire stale routes in the cache, or prefer "fresher" routes when faced with multiple choices. In contrast, AODV has a much more conservative approach than DSR. When faced with two choices for routes, the fresher route (based on destination sequence number) is always chosen. Also, if a routing table entry is not used recently, the entry is expired. The latter technique is not problem-free, however. It is possible to expire valid routes this way if unused beyond an expire time. Determination of a suitable expiry time is difficult, because sending rates for sources, as well as node mobility, may differ widely and can change dynamically. Finally, the route deletion activity using RERR is also conservative in AODV. By way of a predecessor list, the error packets reach all nodes using a failed link on its route to any destination. In DSR, however, a route error simply backtracks the data packet that meets a failed link. Nodes that are not on the upstream route of this data packet but use the failed link are not notified promptly.

TORA is a "link reversal" algorithm that is best suited for networks with large dense populations of nodes [61]. The unique feature of TORA is maintaining multiple routes to the destination so that many topological changes do not require any reaction at all, as having just a single route is sufficient. TORA and DSR are the only on- demand protocols considered here which retain multiple route possibilities for a single source/destination pair. The protocol reacts only when all routes to the destination are lost and hence bandwidth can potentially be conserved because of the necessity for fewer route discoveries. In the event of network partitions, the protocol is able to detect the partition and erase all invalid routes. Another advantage of TORA is its multicast support. Although, unlike AODV, TORA does not incorporate multicast into its basic operation, it functions as the underlying protocol for the Lightweight Adaptive Multicast Algorithm (LAM), and together the two protocols provide multicast capability. If a node does not have GPS or some other external time source, it cannot use the algorithm. Additionally, if the external time source fails, the algorithm will cease to operate. Furthermore, route rebuilding in TORA may not occur as quickly as in the other algorithms, due to the potential for oscillations.

ABR is a compromise between broadcast and point-to-point routing, and uses the connection-oriented packet forwarding approach. Route selection is primarily based on the associativity ticks of nodes along the path. The selected route tends to be more long-lived due to the property of associativity. A long-lived route requires fewer route reconstructions and therefore yields higher throughput. One drawback of ABR, which could affect the latency of route requests, is that intermediate nodes are unable to respond to route requests. The BQ message or LQ message must travel all the way to the destination before a reply is sent. DSR, on the other hand, supports intermediate node replies. In addition, ABR validates only the best routes, duplicate packets are not seen at the destination; consequently, no additional routes to a destination are cached. Furthermore, ABR relies on the fact that each node is beaconing periodically. This beaconing requirement can actually help the network view convergence faster for dynamic topologies. However, this beaconing requirement may result in additional power consumption. The experimental results obtained in [52] reveal that the inclusion of periodic beaconing has a minute influence on the overall battery power consumption.

The SSR algorithm is basically ABR with the additional property of selecting routes based on signal strengths and location stability. As in ABR, while the paths selected by this algorithm are not necessarily shortest in hop count, they do tend to be more stable and long-lived, resulting in fewer route reconstructions. Unlike in AODV and DSR, intermediate nodes cannot reply to route requests sent

towards a destination; this results in potentially long delays before a route can be discovered. In addition, in case of link failure, the route discovery restarts from the source to find a new route to the destination. Unlike ABR, no attempt is made to allow intermediate nodes (partial route discovery) to attempt to rebuild the route themselves. AODV and DSR also do not specify partial route discovering. Since link failure cannot be resolved locally without the intervention of the source node, this may lead to longer route reconstruction times. None of the on-demand protocols supports security or QoS except AODV and DSR. AODV and DSR supports certain quality of service parameters; in particular, Maximum Delay and Minimum Bandwidth.

5.0 Unicast Performance Comparison

When comparing two Ad Hoc networking protocols there are certain metrics, which have been established to be most useful in deciding which ones 'perform' better. These include: measurements of end-to-end data throughput and delay, route acquisition time, percentage of out of order delivery, and efficiency (based on bandwidth overhead). Certain simulators provide the ability to measure some of these metrics better than others. We have found the majority of papers so far have focused on delay and throughput and control overhead, paying less attention to route acquisition time and percentage of out of order delivery. Papers studying only a single protocol will not be considered here for our comparison.

This summary will discuss five papers that performed simulations of ad hoc protocols in order to do a performance comparison. The first paper, Broch et al. [32] at CMU developed mobile networking extensions for the ns-2 network simulator to allow it to simulate wireless networks. Their research then consisted of a performance comparison of AODV, DSR, DSDV, and TORA. Lee et al. [52] compiled performance data on DSR, DBF and ABR, using GloMoSim. The remaining three papers both used the CMU extensions themselves in their own independent simulations. Das et al. [58] compiled performance data on DSR, AODV, TORA and DSDV, Johansson et al. [46] focused on DSR, DSDV, and AODV, while Perkins et al. [7] performed simulation on DSR and AODV.

For the most part the environment parameters used by these papers were very similar. Likely this is in no small part due to the fact that they all used the same simulator. Most of them used a combination of free space and two-ray ground propagation models, a random way-point mobility model, IEEE 802.11 MAC protocol for the link layer, and an ARP implementation modeled after the DBS Unix implementation. In addition, some of them assumed networks of similar size and activity each using approximately 50 nodes, transmitter range of 250 meters, similar roaming areas (1500x300 meters), and constant bit rate streams between nodes, for around 900 seconds of simulation time. Lee et al. considered very different parameters then others. They assumed 50 nodes, transmission range of only 5 meters and a roaming area 20m x 20m. Perkins et al. used a second field configuration. They considered in this configuration 100 nodes, 2200m x 600m roaming area with same transmission range for 500 seconds of simulation time.

The way they modeled the mobility metric did differ slightly. Mobility was modeled by all groups as 'Random Waypoint'. In this model, nodes select random way-points within the roaming area, and travel there at a constant speed randomly chosen from a uniform distribution, U[0, V_{max}]. After reaching its destination, the node waits for some *pause time* then moves to the next waypoint. Both Broch et al. and Das et al. used pause time to define the 'mobility' of the nodes. Zero pause time corresponded to constant motion, while no motion resulted if the pause time was longer than the simulation time. Johansson et al. defined the mobility metric as the V_{max} of the uniform speed distribution, in general keeping the pause time constant. To what extent these differences would affect the outcome is not known. It is likely that the only difference will be an inconsistent scale, since in both cases the maximum speed for any node was 20m/s, and this minimum speed was 0.

Traffic load was also not completely consistent. Johansson and Broch both used 64 byte packets, while Das and Perkins used 512 byte data packets. Johansson took data at varying packet send rates (from 5 to 20 packets/s) in order to vary the traffic load metric. By contrast, Broch and Das kept all their simulations at 4 packets/second and increased the number of CBR sources in order to vary the traffic load metric. Perkins kept data rate at 4 packets/second for 10, 20 and 30 sources and 3 packets/second for 40 sources for the 50 nodes experiment. For 100 nodes experiment, kept data rate 4 packet/seconds for 10 and 20 sources and 2 packets/second for 40 sources.

Johansson's paper also conducted a series of simulations in what they described as realistic scenarios, including: Conference, Event Coverage, and Disaster Relief These scenarios involved modified propagation models to include primitive shadowing, as well as choosing average speed, and node positions to correspond to the scenario. The conclusions they reached from those scenarios were the same as those reached from the rest of their analysis, so the specifics will not be discussed below.

5.1 Results Comparison

Each group measured slightly different performance metrics from the simulation. Johansson and Lee measured the end-to-end delay, the throughput, and the routing overhead of the system [46, 52]. Broch measured the packet delivery ratio, the routing overhead, and path optimality [32]. Das measured the packet delivery ratio, the end-to-end delay, and the routing overhead [58]. Perkins measured packet delivery fraction, average end-to-end delay of data packet, normalized routing load and normalized MAC load [7]. How each of these performance metrics are affected by the protocol used, the traffic load, and the mobility will be discussed in turn.

5.1.1 Routing Overhead

It was generally agreed that as mobility of the network increases, the routing overhead increases as well. The lowest overhead in general was exhibited by DSR. It was generally agreed that the reason was that it had fewer control packets to send, since it does not require sending periodic updates. AODV was slightly worse since it also required these periodic updates to be sent. DSDV's overhead was constant with respect to the mobility, a credit to the number of triggered updates being sent through the network. That was a hindrance at low mobility though where the extra route checking was not needed.

The relative performance of TORA and Idealized Link-State (ILS) protocol was found to be critically dependent on the network size and the average rate of topological changes [60]. The results indicated that for a given available bandwidth- as the network size and/or rate of topological change increases, the amount of control overhead for ILS increases much more rapidly than for TORA - effectively congesting the communication channel and causing additional queuing delay for message traffic. Based on simulation results by Broch et al., it is best suited to large, densely packed arrays of nodes with very low node mobility. Of the two papers reporting simulation results for TORA, only the one by Broch et al. actually simulates node mobility. Broch found TORA to be encumbered by its layering on top of IMEP, and found IMEP caused considerable congestion when TORA was trying to converge in response to node mobility. This resulted in TORA requiring between one to two orders of magnitude more routing overhead than other ad hoc routing protocols investigated by Broch et al.

As ABR sends beacon messages to maintain the list of neighbors, one might expect ABR to have considerably more control overhead when nodes are stable. However, result shows only a small difference since the size of a beacon message very small. Result also shows that increasing the mobility speed makes ABR more efficient.

Increasing the network load generally increased the overhead required for any protocol. DSDV remained almost constant with respect to load. DSR responded the next to increased load, remaining fairly low. AODV faired the worst. One should note however, that those results are when overhead is measured in terms of control packets sent. When measuring in terms of actual bytes sent, DSR performs much worse than AODV. This is because of the size of the headers of each of the DSR packets with respect to the data packet size.

5.1.2 Packet Delivery Ratio

As might be expected, the fraction of packets delivered generally decreased with node mobility. In this contest AODV and DSR were almost equal, a small difference became noticeable when the traffic load on the network was also increased.

Interestingly, Das, Perkins and Broch saw different protocols do better under those conditions. It is possible that because Das and Perkins were using much larger data packets, and DSR has a higher per packet overhead, that higher packet loss occurred due to node transmission queues overflowing more often when DSR was used. It must be examined further to be sure. DSDV operated only slightly worse at low mobility, but saw a dramatic decrease in delivery ratio when the pause time decreased below 300 seconds. It is believed that this was due to the lack of path redundancy in DSDV, so that a broken path was disproportionately more difficult to fix in high mobility environments.

5.1.3 End-to-End Delay

Predictably, delay was more pronounced at high loads and high mobility. DSDV enjoyed the best performance by this metric. Because it maintains shortest path routes, when packets are transmitted successfully, they get through quickly. DSR might be expected to lag slightly due to the extra time required to examine each packet header. In spite of this, AODV and DSR were evenly matched at low loads and low mobility. At high loads Das reported AODV performing with lower delays, while Johansson's data indicated negligible differences still. No explanation for this discrepancy is immediately available. Perkin's result shows the delays for DSR are larger than AODV by a factor of about 2-6 for high mobility, with the factor increasing with the number of sources.

Perhaps less expected was the fact that at high loads AODV and DSR enjoys a lower delay for moderate mobility and low mobility. This was apparently due to moderately mobile nodes breaking routes before their internal queues became full. This was compared to a sort of automatic load balancing occurring within the node cluster.

5.1.4 Path Optimality

Broch's paper used an internal mechanism in the simulator to calculate the shortest path between nodes, and compared that to the actual number of hops taken by a packet. In general they found that DSDV and DSR frequently use the most optimal routes, while AODV took as many as 4 additional hops to reach the destination. The optimality of AODV's routes was also inversely proportional to mobility, using more optimal routes when mobility is low.

5.1.5 Throughput

Johansson's group found that DSR and AODV are basically evenly matched when it came to throughput and responded very little to increased mobility. DSDV, on the other hand, saw a large drop (as much as 40 percent) in throughput as mobility was increased. As the load increased, the throughput predictably increased as well. AODV responded the best, strangely followed by DSDV. At high loads DSR had difficulty maintaining throughput for increasing mobility, dropping by as much as 40 percent. This was attributed to DSR packets containing the entire source route, so they incur larger sizes and higher processing times at each node.

Lee's [52] result shows that the throughput performance of ABR is very well when nodes are static but decreases gradually with increasing node mobility. Toh et al. [11] introduced a dynamic cell size adjustment scheme based on the methodology of excluding inactive neighbors by reducing one's transmission power. This reduction in radio coverage basically lessens the number of contenders in a wireless cell, hence increasing the transmission throughput in a manner, which is essentially independent of the underlying MAC layer protocol

5.2 Discussion

Based on these previous analyses several conclusions can be drawn. In general DSDV performs well in situations with low to zero mobility. Depending on the measure of performance it can even outperform the others, but only at zero mobility. Ad hoc networks, however, demand that performance does not suffer greatly due to high mobility, so it should not be considered as a viable option in most applications. TORA suffers from very high overhead. In ABR, sending route updates periodically and triggering updates when the topology changes in order to maintain an up-to-date routing table results in excessive control message overhead, which is unacceptable in a wireless environment with limited bandwidth. The throughput also decreases with mobility. Therefore, TORA and ABR also should not be considered as a viable option. Deciding whether DSR or AODV performs better is a much more daunting task. The data regarding the performance of the two is not yet consistent. There is evidence that where overhead is a concern AODV performs better than DSR. In other areas though agreement is low, and variability in the data is high. Further research much be done before the superiority of one over the other can be determined.

6.0 Existing Multicast routing Protocol

Sending a message to well-defined groups that are numerically large in size but small compared to the network as a whole is called multicasting, and its routing algorithm is called multicast routing. Ad hoc networks are interesting to a large extent because of the challenge of maintaining a communication path between a source and a destination, even when some of the intermediate forwarding nodes are unable to continue participating in packet forwarding and must be replaced by nodes along another path. It turns out that maintaining paths between a single source and multiple destinations is somewhat more difficult, but not excessively so. Given the growing importance of multicast as a means to reduce the bandwidth utilization for mass distribution of data, and the pressing need to conserve scarce bandwidth over wireless media, it is natural that multicast routing should receive some attention for ad hoc networks.

There are three categories of multicast routing protocols. Proactive, protocols pre-computes paths to all possible destinations and stores this information in routing tables. To maintain an up-to-date database,

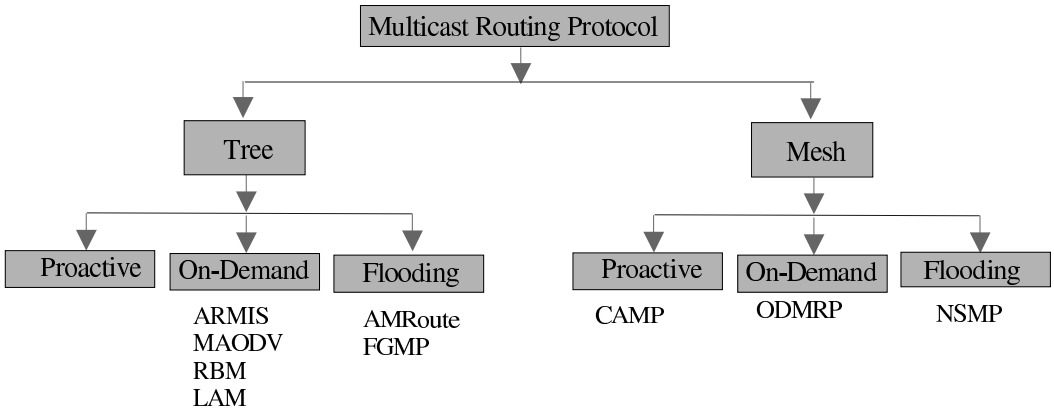


Figure 39. Categorization of Ad-Hoc Multicast Routing Protocols

routing information is periodically distributed throughout the network. An example is CAMP. Reactive or on-demand, protocols create paths to other hosts on-demand. The idea is based on a query-response mechanism or reactive multicast. In the query phase, a node explores the environment. Once the query reaches the destination, the response phase is entered and establishes the path. Examples are ODMRP, MAODV etc. Lastly, simply flood the network. Every node receiving a message floods it to a list of neighbors. An example is AMRoute.

6.1 Adhoc Multicast Routing Protocol (AMRoute)

The Adhoc Multicast Routing Protocol (AMRoute) [25] allows for robust IP Multicast in mobile ad hoc networks by exploiting user-multicast trees and dynamic cores. AMRoute emphasizes robustness even with rapidly changing membership or highly dynamic networks; it does not attempt to provide the absolute minimum bandwidth or latency guarantees in a given topology. The two key new features of AMRoute that make it more robust and efficient in ad-hoc networks are:

- User-multicast trees, where replication and forwarding is only performed by group members over unicast tunnel.
- Dynamic migration of core node according to group membership and network connectivity.

The user-multicast tree includes only the group senders and receivers as its nodes. Each node is aware of its tree neighbors only, and forwards data on the tree links to its neighbors. Multicast state is maintained by the group nodes only, and is not required by other network nodes. In fact, AMRoute does not even

require non-member nodes to support any IP multicast protocol. The elimination of state in other network nodes clearly saves node resources, especially when compared with broadcast-and-prune native multicast protocols that require per source and per group state at all network nodes. More importantly, especially in highly dynamic ad hoc networks, user-multicast trees also eliminate the need to change the tree as the network changes. Neighboring tree nodes are inter-connected by IP-in-IP tunnels. Consequently, assuming unicast connectivity is maintained among member nodes, the AMRoute distribution tree will continue to function despite network changes. Each group in the network has at least one logical core that is responsible for discovering new group members and creating/maintaining the multicast tree for data distribution. This logical core is not a preset node (chosen from among currently known members) and it can change dynamically. The absence of a single point of failure is an important requirement for adhoc networks.

AMRoute includes some of the best features of other multicast protocols. Like CBT and PIM-SM it uses shared-trees, so only one tree is required per group, improving its scalability. Similar to some other multicast protocols (DVMRP, PIM-DM etc), AMRoute is independent from specific semantics of the underlying unicast routing protocol. Tying the protocol closely with a unicast protocol some efficiency gains is possible but still the independence as more important. Its independence allows use of the optimal ad-hoc unicast protocol for the network and can work transparently across domains supporting different unicast protocols. It provides robustness by periodic flooding for tree construction. Instead of flooding data, AMRoute periodically floods a small signaling message.

6.1.1 Assumptions (or lack thereof)

AMRoute assumes the existence of an underlying unicast routing protocol that can be utilized for unicast IP communication between neighboring tree nodes. However, no assumptions are made about the syntax or semantics of the unicast protocol, and the same unicast protocol is not required to be used in all domains. The actual paths followed by the two directions of a unicast tunnel connecting neighboring group members may be different. It is not required that all network nodes support AMRoute or any other IP Multicast protocol. Non-multicast routers only need to support unicast, and forward the control packets (without any protocol processing) if the IP-level TTL is non-zero during the expanding ring search.

Both multicast receivers and senders are required to explicitly join the multicast tree and perform data forwarding, which is slight departure from the classical IP Multicast model. All members (senders and receivers) must be capable of processing IP-in-IP encapsulation. No group members need to have more than one interface or act as unicast routers. However, at least one member, and ideally all nodes, must be capable of being AMRouters: forwarding and replicating datagrams to other members. AMRoute routers must be able to set the TTL field in the IP headers (decrementing the inner IP TTL field before a datagram is repackaged and forwarded).

We assume that the diameter of an ad-hoc network will be limited. Packets may be lost or corrupted in transmission on the wireless network.

6.1.2 Concepts

AMRoute creates a per group multicast distribution tree using unicast tunnels connecting group members. The protocol has two key parts: mesh creation and tree creation. Only logical core nodes initiate mesh and tree creation; however, the core can migrate dynamically according to group membership and network connectivity. Bi-directional tunnels are created between pairs of group members that are close together, thus forming a mesh. Using subset of the available mesh links, the protocol periodically creates a multicast distribution tree.

6.1.2.1 User-Multicast Distribution Trees

Figure 40 shows a user-multicast tree connecting six members. The group members forward and replicate multicast traffic along the branches of the virtual tree. The datagram sent between logically neighboring

nodes is physically sent on a unicast tunnel through potentially many intermediate routers. The path taken by the unicast tunnel can change without affecting the User-multicast tree.

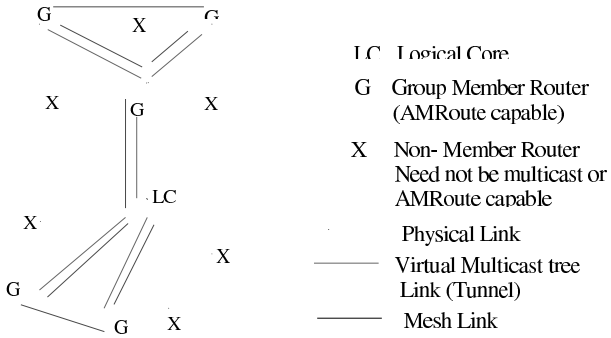


Figure 40. A Virtual User-Multicast

There are two key advantages to implementing multicast protocol using only members:

- The user-multicast tree need not change because of node movement, provided there remains a path among all nodes connected by mesh branches. This independence improves robustness and reduces signaling overhead.
- Non-members are not required to perform packet replication or even support any IP multicast protocol. This places the processing and storage overhead needed to support the multicast tree only on members.

The penalty paid for using a user-multicast tree is reduced efficiency. Since non-member routers are not allowed to perform packet replication, the bandwidth efficiency is reduced. In addition, the delay is often increased. However, the difference between an optimal tree using all network nodes and an optimal tree using only member nodes is theoretically bounded.

In a stable network, the worst case bandwidth overhead for forwarding on a user multicast tree is less than twice as that for an optimal tree using all network nodes. This worst case overhead occurs in case of a simple star network with a non-member central node directly connecting n member nodes: a tree using the central node will take n-hops, while a user multicast tree will take 2* (n-1) hops. The worst case delay overhead depends on the time it takes to do packet replication. Assuming all nodes can only send one packet at a time, and then the worst case delay overhead is again twice as much as the other case. (However, if a router can send on multiple interfaces simultaneously, then the worst case delay overhead will be higher).

In a dynamic network, the overhead becomes more complex to calculate. The bandwidth overhead for using an optimal tree must include the signaling overhead of maintaining the tree. On the other hand, a user multicast tree has the overhead of the tree becoming less optimal as nodes move. The balance between these overheads depends on the mobility pattern.

Just as there are many ways to create native multicast trees, there are many ways to create user-multicast trees. We now describe a way that is well suited to ad-hoc networks.

6.1.2.2 Logical core and non-core members

In the AMRoute protocol each group has at least one logical core that is responsible for initiating signaling actions, specifically: a) mesh joins (discovering new group members and disjoint mesh segments and b) multicast tree creation. A non-core node can act as a passive responding agent and cannot initiate these two operations. Limiting the number of nodes that perform these two functions (ultimately to a single logical core) ensures that AMRoute can scale, without causing excessive signaling or processing overhead.

The AMRoute logical core node avoids robustness problems while permitting scalability since it:

- is not a central point for all data. Forwarding can continue on working branches of the tree irrespective of the status of the logical core and links to the logical core.
- is not a preset node. Each multicast tree segment designates one of its nodes to be the core based on the core resolution algorithm.
- changes dynamically. The core node migrates according to group membership and network connectivity.

An AMRoute segment can temporarily have more than one core node for a group after new nodes join or disjoint segments merge. A network node designates itself as a core when first joining a group. As a logical core a node can quickly discover new group members and join the mesh and tree with its closest neighbors (not just to the existing core). When multiple core nodes exist in a segment, they will advertise their presence by sending out tree creation messages. Core nodes use the reception of tree creation messages from other cores to decide whether to remain as a core. The core resolution algorithm is run in a distributed fashion by all nodes. The goal of the algorithm is to ensure that any group segment has exactly one core node and that the core node migrates to maintain a robust and efficient multicast tree.

Since the core node sometimes disappears (e.g., leaves the group) or an existing segment is split into multiple disjoint segments (e.g., because of link or node failure), an AMRoute segment can also have no core nodes. If a segment does not have a core node, one of the nodes will designate itself as the core node at some random time, on not receiving any tree creation or join messages.

A key issue with any algorithm that assigns a single core node is that it can centralize the multicast tree and indeed the mesh links on itself. AMRoute prevents centralization in a number of ways:

- A non-core node is not allowed to graft to its own logical core. Without this limitation all group members would ultimately be connected to the core.
- All nodes, including the core, are only allowed to have a limited number of tree links. If the limit is reached the node must drop the link furthest (at highest cost) from its current location.
- A logical core will only take responsibility as core for a limited time or until some event makes changing the core desirable. A new logical core can be picked, for example when the core's mesh connectivity limit is reached.

Clearly the core resolution and change algorithms are key to the robustness and performance of the AMRoute protocol. However, it is also desirable to contain the complexity of the algorithms.

6.1.3 Operation

We now discuss the operation of AMRoute, with an emphasis on explaining the design choices. The detailed design is available in the AMRoute specification.

6.1.3.1 Control Message

AMRoute uses five control messages for signaling purposes. These are sent directly over IP. JOIN_REQ is the only message broadcast using an expanding ring search, while the other control messages are unicast between group members. The data packets are sent over UDP/IP, with no separate encapsulation for AMRoute. The message formats can be found in the AMRoute design specification. Their purpose can be summarized as follows:

- JOIN_REQ: This is broadcast periodically by a logical core for detecting other group segments.
- JOIN_ACK: This is generated by a tree node in response to receiving a JOIN_REQ from a different logical core.
- JOIN_NAK: This is generated by a tree node if its application leaves the group.
- TREE_CREATE: This is generated periodically by a logical core to refresh the group spanning tree.
- TREE_CREATE_NAK: This is generated by a tree node to convert a tree link into a mesh link.

A group member uses a monotonically increasing sequence number for unicast control messages generated for each group. These are used by its tree neighbors for ordering of received control messages, so that older messages are not processed after newer one have been received.

6.1.3.2 Mesh Creation

An AMRoute mesh is a graph where each node is a member (sender or receiver) of the group and every link is a bi-directional unicast tunnel. While the mesh establishes connectivity across all the members of a group, a tree is needed for forwarding the data. We use a two step process of creating a mesh before the tree because it is simpler and more robust.

A mesh is much simpler to maintain (that could potentially have loops) than a tree (with no loops) at every stage of member mutual discovery phase. For example, a very naive merging algorithm could result in a loop when three disjoint trees discover each other. In addition, the redundant mesh links contribute towards increased robustness in the case of node/link failures. (Note that the use of unicast tunnels between neighboring nodes of the mesh itself contributes towards robustness in the face of intermediate node/link failures along routes between them as the unicast protocol is expected to establish a separate route around the failed network node/link).

6.1.3.3 Joining and Leaving a Group

The expanding ring search mechanism based on TTL-limited broadcasts is adopted to allow members to discover each other. All core nodes periodically broadcast JOIN-REQ messages. The period between JOIN-REQ will be proportional to the TTL value associated with the last JOIN-REQ message.

Each member begins by identifying itself to be the core of a 1-node mesh consisting of only itself. The core node sends out JOIN-REQ packets with increasing TTL to discover other members of the group. When a member (core or non-core) node receives a JOIN-REQ from a core for a different mesh for the same group, the node responds back with a JOIN-ACK. A new bi-directional tunnel is established between the core and the responding node of the other mesh. As a consequence of mesh mergers, a mesh will have multiple cores. One of the cores will emerge as the winning core of the unified mesh as a result of the core resolution algorithm.

If a node leaves a group, it sends out a single JOIN-NAK message to its neighboring nodes. If it subsequently receives any data or signaling message for that group it can send out further JOIN-NAK messages.

6.1.3.4 Becoming a Passive Non-core node

Only logical core nodes initiate the discovery of other disjoint meshes, while both core and non-core nodes respond to the discovery messages. It is simpler and more scalable (in bandwidth usage) to have only the core node initiate discovery as against every node of the mesh. However, to avoid the situation where every merger adds a link to a core (which might result in too many links from the core), non-core nodes can participate in the mergers by responding to discovery messages received from other cores.

6.1.3.5 Limitation on link connectivity

A node must break one or more of its links, whenever, the number of links adjacent to a node exceeds LINK-THRESHOLD. Each of the links could be associated with a weight representing the distance (example, number of hops) from the neighbor. The links chosen to be broken could be the ones with the farthest neighbors. The farthest neighbor is notified about the link breakage using a JOIN-NAK message. If the message is lost and data is received from this non-neighbor, the JOIN-NAK message will be resent.

To maintain a reasonably optimal mesh in the face of mobility, periodic mesh reconfiguration is required; however, removing links may result in temporary loss of data and additional overhead. When the links

are broken, the mesh might be fragmented into disjoint meshes. Fragmentation is handled in the same manner as node failures.

6.1.3.6 Dynamic Core Migration

There might be a need for dynamically migrating the logical core so as to make the tree more optimal. If the core is closer to the senders, then the tree may be a close approximation of a source-based tree (which is shared). If the core is at the center of the mesh, then TREE-CREATE messages reach all the nodes of the mesh faster than when the core is at the edge. In addition, as the core is involved in discovering and merging with other disjoint meshes, dynamically changing the core might help in avoiding the situation where there are excessive links adjacent to a core node. Policies and mechanisms for node changes are still under research.

6.1.4 Tree Creation

This section discusses the creation of a tree for data forwarding purposes once a mesh has been established. The core is responsible for initiating the tree creation process. From the point of view of individual nodes of
the mesh, this phase involves identifying the subset of links adjacent to it that belong to the tree.

6.1.4.1 Periodic TREE-CREATE Broadcast

The core sends out periodic TREE-CREATE messages along all the links adjacent to it in the mesh. (Note that TREE-CREATE messages are sent along the unicast tunnels in the mesh and are processed by group members only, while JOIN-REQ messages are broadcast messages that are processed by all network nodes). The periodicity of the TREE-CREATE messages depends on the size of the mesh and also on the mobility of the nodes of the mesh. The number of hops between neighbors keeps changing dynamically as the mesh nodes are mobile. Thus, when TREE-CREATE messages are sent out, newer and more optimal trees might be created. Group members receiving non-duplicate TREE-CREATEs forward it on all meshes links except the incoming, and mark the incoming and outgoing links as tree links. If a node has a collection of neighbors all 1-hop away on the same broadcast capable interface, then the node can send a single broadcast message to all 1-hop neighbors simultaneously.

6.1.4.2 Purging of Tree Links

If a link is not going to be used as part of the tree, the TREE-CREATE message is discarded and a TREE-CREATE-NAK is sent back along the incoming links. On receiving a TREE-CREATE-NAK, a group member marks the incoming link as a mesh link and not a tree link. Thus each non-core node considers the link along which a non-duplicate TREE-CREATE message was received and every other link along which no TREE-CREATE-NAK message was received to be part of the tree for a specific group. (Core considers every link adjacent to it to be part of the tree). Note that all these tree links are bi-directional tunnels.

The choice of using ACK or NAK in response to the TREE-CREATE messages is dictated by whether robustness or saving bandwidth is more important. If a ACK-based (positive acknowledgment) scheme is used, then data may not be delivered along links where ACKs were lost. However, if a NAK (negative acknowledgment) based scheme is used, loss of NAKs can only result in the same data being forwarded more than once (which is discarded by the downstream node on reception).

When data arrives at a node along one of the tree links, the node forwards the data along all other tree links. However, if data arrives along a non-tree link a TREE-CREATE-NAK message is (again) sent back along that link and the data is discarded.

6.1.4.3 Transient Loops

The tree created by the nth TREE-CREATE message might not be the same as the one created by (n-1)th message. A situation may exist where some nodes are forwarding data according to the older tree and some according to the newer tree, which may result in loops or data loss. Such a phase is to be expected due to the dynamic nature of ad hoc networks. However it is considered to be transient and AMRoute recovers from it as soon as the network reduces its dynamicity.

6.1.4.4 Effect of Node Failures and Group Leaves

Node failures or Nodes leaving a group are only partially handled by the redundant links in the mesh. In some situation, node failures might result in splitting the mesh into multiple disjoint meshes, where only one of these meshes has the core. Each node in the mesh expects to periodically receive TREE-CREATE messages. In case this message is not received within a specific timeout, the node designates itself to be the core after a random time. The node whose timer expires the earliest succeeds in becoming the core and initiates the processes of discovering other disjoint meshes as well as tree creation. Multiple cores that may arise in this case are resolved by the core resolution procedure.

6.1.4.5 Core Resolution

In case of mesh mergers, there might be multiple active cores in the new mesh. When nodes in the mesh receive TREE-CREATE messages from multiple cores, then they become aware of this situation. To decide on a unique core for the mesh, the nodes execute a core resolution algorithm and forward TREE-CREATE messages arriving from the unique core and discard TREE-CREATE messages from other cores. As the multiple cores in the mesh will also become aware of the existence of other cores, they will also execute the same core resolution algorithm. All the cores except the *winning* core will demote themselves to non-core state. One simple core resolution algorithm could pick the winning core to be the one with the highest IP address.

6.1.4.6 Picking which branch to use for the Tree

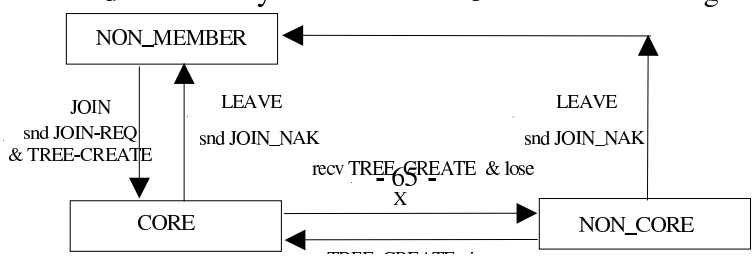
The simplest approach for picking a mesh-branch as a tree-link is to simply accept the first TREE-CREATE message that is received, and to discard any duplicate TREE-CREATE messages using the sequence number included in each TREE-CREATE message. This results is a reasonable tree, but it is not necessarily the most bandwidth efficient (e.g., using minimum number of total hops) or lowest latency. Further research for investigation of tradeoff of increasing the complexity of branch selection in order to improve bandwidth efficiency or reduce latency is required.

To improve bandwidth efficiency a node can select the branch from which it received a TREE-CREATE from its closest neighbor (based on the TTL value in the outer IP header). However, in order to prevent the tree from becoming broken (not connecting all group members), the algorithm must a) be able to tell when the TREE-CREATE message has been received before, and b) not change the initial selection until the next round of TREE-CREATE messages are received. To detect duplicates it necessitates the use of a path-vector field in the TREE-CREATE message

6.1.4.7 State Diagram

Figure 41 shows the three main AMRoute states and state transitions (with causing events and resulting actions). The states can be interpreted as follows :

- NON-MEMBER - a node does not belong to the multicast group.
- CORE - a node currently recognizes itself to be a logical core.
- NON-CORE - a node is currently a non-core member in the multicast group.



A node transitions from the NON-MEMBER state when an application on the node joins a group and transitions to it from all other states when the application leaves the group.

A node transitions to the CORE state when an application joins a group, and by default sets itself to be a logical core. A logical core sends out periodic JOIN-REQ messages and TREE-CREATE messages. A logical core becomes a non-core node if it loses in the core resolution procedure that ensues when it receives a TREE-CREATE message from another core belonging to the same multicast group, which means the other core becomes the new core.

A non-core member expects periodic TREE-CREATE messages from a logical core. If it does not receive one within the specified period, the associated timer expires and the node resets itself to be a core.

6.1.4.8 Timers

A logical core keeps two timers, namely the JOIN-REQ-SEND timer and the TREE-CREATE-SEND timer. The expiry of JOIN-REQ-SEND causes the node to compute the new TTL value to use for the expanding ring search, broadcast a new JOIN-REQ with this TTL value and reset the timer. The TREE-CREATE-SEND timer is kept to send out periodic TREE-CREATE messages.

A non-core member uses a TREE-CREATE-RCV timer. When it expires, the node waits for some random amount of time before it resets itself to be a core, and starts sending out JOIN-REQs and TREE-CREATEs. This period is set to be random to prevent multiple non-core nodes from becoming cores at the same time.

6.1.4.9 Data Structures

Each member keeps two tables, each containing a set of neighbors, the neighbors on the mesh and the neighbors on the multicast tree. These neighbors are connected by unicast tunnels rather than being physical neighbors. The member also keeps a *hop-count* associated with each mesh link. This hop-count is obtained at the time of mesh link creation, and is updated by the periodic control messages (control messages have original TTL value used by source in the headers).

A node tracks the ID of the logical core it currently recognizes, which can be itself. Multiple logical cores can exist in a same group. However, each node only recognizes one logical core at any instant. This information is updated when the node receives a TREE-CREATE from a logical core it did not recognize, or when the node resets itself to be core.

6.1.5 Strengths and Weaknesses

Strengths:

- Provided there remains a path among all nodes connected by mesh branches, the user-multicast tree need not change because of network changes (e.g., because of router movement). This independence improves robustness and reduces signaling overhead.
- Non-members are not required to perform packet replication or even support any IP multicast protocol. This places the processing and storage overhead needed to support the multicast tree only on members.

- AMRoute include some of the best features of other multicast protocols. Like CBT and PIM-SM, it uses shared-trees, so only one tree is required per group, improving its scalability.
- AMRoute uses unicast tunnels as tree links to connect neighbors on the user-multicast tree. Thus, AMRoute does not need to be supported by network nodes that are not interested/capable of multicast, and group state cost is incurred only by group senders and receivers. AMRoute does not require a specific unicast routing protocol; therefore, it can operate seamlessly over separate domains with different unicast protocols.

Weaknesses:

- The penalty paid for using a user-multicast tree is reduced efficiency. Clearly, bandwidth efficiency is reduced as non-member routers are not allowed to perform packet replication.
- The delay is often increased. It can be shown, however, that there is at most a doubling in bandwidth needs and increase in delay.
- As the network becomes more dynamic, maintaining a more optimal path will become harder and require increasingly higher signaling overhead.

6.2 Multicast Ad Hoc On-Demand Distance Vector Protocol (MAODV)

The Multicast Ad Hoc On-Demand Distance Vector (MAODV) Protocol [21] is capable of multicast communication. Multicast routes are discovered on-demand and use a broadcast route discovery mechanism. It offers quick adaptation to dynamic link conditions, low processing and memory overhead, and low network utilization. MAODV creates bi-directional shared multicast trees connecting multicast sources and receivers.

6.2.1 Route Tables

The following information is stored in each entry of the multicast route table.

- Multicast Group IP Address, Multicast Group Leaders IP Address
- Multicast Group Sequence Number, Next Hops, Lifetime
- Hop Count to next Multicast Group Member, Hop Count to Multicast Group Leader

The Next Hops field is a linked list of structures, each of which contains the following fields:

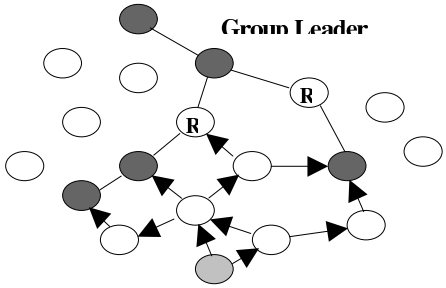
- Next Hop IP Address, Next Hop Interface, Link Direction and Activated Flag.

Nodes may also maintain a Group Leader Table. The fields of the group leader table are:

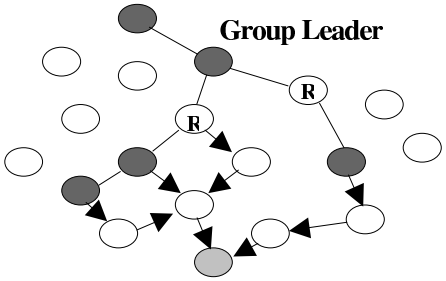
- Multicast Group IP Address and Group Leader IP address.

6.2.2 Route Discovery

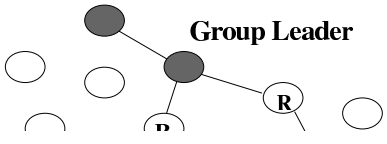
A node wishing to join a multicast group broadcasts a RREQ with the destination IP address set to that of the multicast group and with the 'J'(join) flag set to indicate that it would like to join the group. If a node receives a join RREQ for a multicast group of which it is not a member, or if it receives a RREQ and does not have a route to that group, it creates a reverse route entry to the source and then broadcasts the RREQ to its neighbors. Figure 42 (a) shows the propagation of a join RREQ.

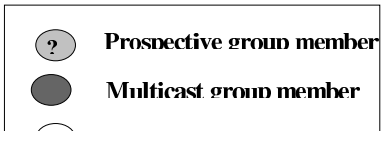


(a) RREQ message propagation



(b) RREPs sent back to source





The source node waits the length of the discovery period to receive a reply. If it does not receive a reply within that time, it rebroadcasts its request with the broadcast ID increased by 1. It continues to do this until it either receives a reply or sends the RREQ rreq_retries additional times. If no reply is receives after this maximum number of broadcasts, it can be assumed that no other group members exist in the connected portion of the network. If the node was attempting to join the group, it becomes that group's leader.

If the associated Activated flag in the multicast route table is false, the node does not forward any data packets for the multicast group along that link. Only after the link is enabled can it be used to send data packets.

6.2.3 Forward Path Setup

If a node receives a join RREQ for a multicast group, it may reply if it is a router for the multicast group's tree and if its recorded sequence number for the multicast group is at least as great as that contained in the RREQ. The responding node updates its multicast route table by placing the requesting node's next-hop information in the table and then generates a RREP. The node unicasts a RREP back to the node indicated in the RREQ. Figure 42 (b) illustrates the path of the RREPs to the source node.

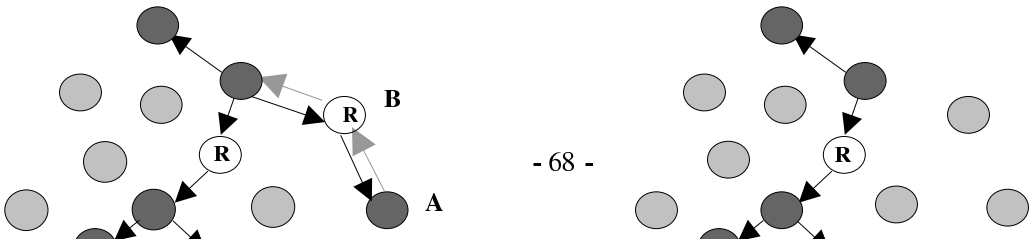
As nodes along the path to the source node receive the RREP, they set up a forward path entry for the multicast group in their multicast route table by adding the node from which they received the RREP as a next hop. Then they increment the Hop Count field and forward the RREP to the next node.

6.2.4 Multicast Route Activation

During the discovery interval, the source node keeps track of the route with the greatest multicast group sequence number and the smallest hop count to the multicast tree. At the end of the discovery interval, it activates that route by unicasting a multicast activation (MACT) message to its selected next hop and by setting the Activated flag for that entry in its multicast route table. Once the next hop receives this message, it activates the route, if it was not the originator of the RREP, then sends its own MACT message to its next hop. This continues until the originator of the RREP is reached. At that point, the path to the multicast tree has been determined. For a join request this branch has been successfully added to the multicast tree and can be used for forwarding data packets to the new multicast group member. Figure 42 (c) illustrates a multicast tree branch addition created in this manner.

6.2.5 Multicast Route Deactivation

A leaf node that wishes to revoke its member status unicasts a MACT message with the Prune flag set to its next hop. It then deletes the multicast group information from its multicast route table. When the next hop receives the prune message, it deletes the next-hop information for the sending node. If the deletion



of this next-hop entry makes this node a leaf, and if the node is not a multicast group member, it may similarly prune itself from the tree by unicasting a prune message to its next hop. Otherwise, if it is still not a leaf node or if it is a member of the multicast group, it does not prune itself from the tree.

Figure 43 illustrates the removal of a multicast group member from the multicast tree. In Figure 43 (a), node A decides to leave the group and sends its upstream neighbor, node B, a MACT message with a set Prune flag. When node B receives the MACT message and deletes node A from its list of next hops, it discovers that it is now a leaf node itself. Because it is a router for the tree and not a group member. Figure 43 (b) shows the multicast tree after B prunes itself as null.

6.2.6 Multicast Tree Maintenance

Each multicast link requires ongoing route maintenance to ensure that other multicast tree members are always reachable. Multicast tree maintenance takes two forms: repairing a broken tree branch following a link break, and reconnecting the tree after a network partition.

6.2.6.1 Link Breaks

A link break is detected, if no data packets have been sent recently, a node must receive a broadcast from each of its next hops at every *hello_interval*. This broadcast can be a RREQ, a GRPH (Group Hello), or multicast data packets. A Hello message is a form of unsolicited RREP, with a TTL of 1. If a node has not broadcast anything within the last *hello_interval*, it must broadcast a Hello so that its next hops on the multicast tree know that it is still within transmission radius. Failure to receive any broadcasts from a next hop on the multicast tree for *hello_life* [(1+allowed_hello_loss)*hello_interval] indicates that the next hop is out of transmission range and so link must be repaired.

When a link break occurs, the node downstream of the break is responsible for repairing it. This distinction is made to avoid the formation of loop. The downstream node initiates the repair by broadcasting a join RREQ for the multicast group which includes an extension field indicating the sending node's distance from the group leader. Only nodes on the multicast tree that are at least this close to the multicast group leader may reply to the RREQ. This prevents nodes on the same side of the break as the initiating node from responding, thereby ensuring that a new route to the group leader is found.

Because the node with which the initiating node lost contact is likely to still be nearby, the initial TTL of the RREQ is set to a small value. In this way, the effects of the link break can be localized. If no RREP is received within the local discovery period, all successive RREQs are broadcast across the network. Any node can respond to a RREQ by sending a RREP as long as it satisfies the following conditions:

- It is a part of the multicast tree.

- It has a fresh enough multicast group sequence number.
- Its hop count to the multicast group leader is smaller than that indicated by the extension field of the RREQ.

In Figure 44 (a), the node indicated as downstream initiates the repair by broadcasting the RREQ. In this example, it sets the extension Hop Count field to 2 because it is two hops from the group leader.

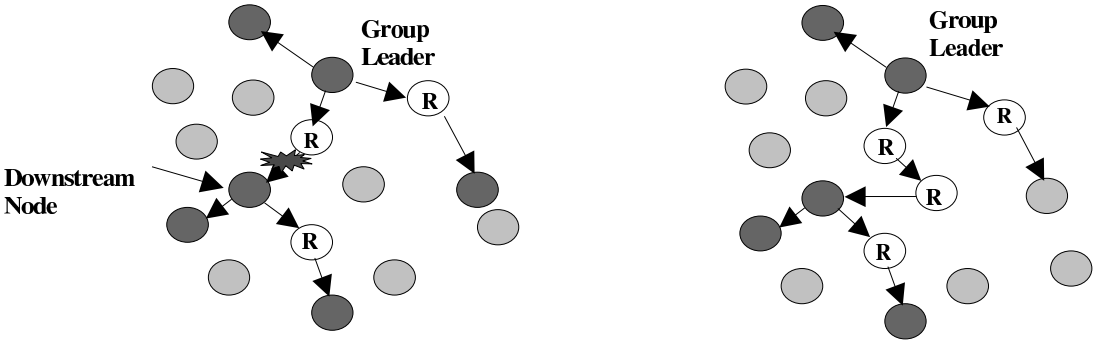


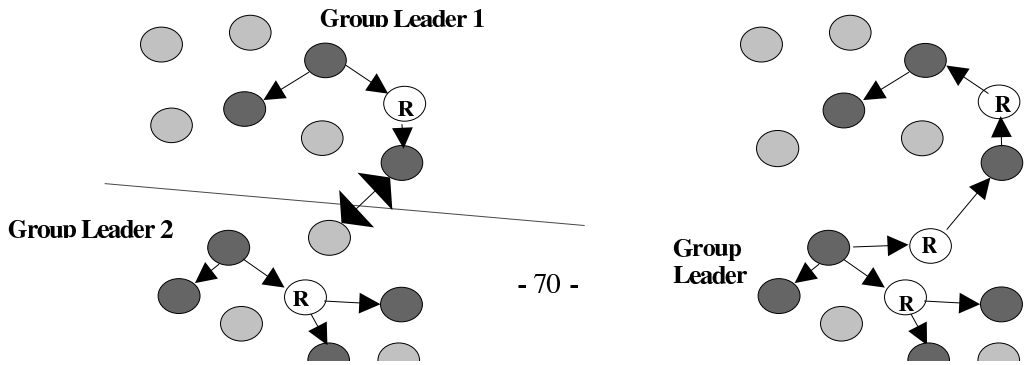
Figure 44. Repair of a Broken Tree Link

Since the node was repairing the tree break, it is possible that it is now a different distance from the group leader than it was before the break. If this is the case, it must inform its downstream next hops of their new distance from the group leader by broadcasting a MACT message with the Update flag set and the Hop Count field set to the node's new distance. This Update flag indicates that multicast tree nodes should update their distance. If these nodes have downstream next hops, they must send them a MACT message with a set Update flag, and so on. The Hop Count field is incremented by 1 each time the packet is received. Figure 44 (b) illustrates the multicast tree after the repair has been completed. If the node initiating the repair does not receive a RREP after rreq_retries additional attempts, it can be assumed that the network has become partitioned and that at this time the tree cannot be repaired. Thus the tree that is downstream from the break is left without a group leader, a new group leader must be selected and can be in the following ways. If the node that initiated the route rebuilding is a multicast group member, it becomes the new multicast group leader. On the other hand, if it was not a group member and has only one next hop for the tree, it prunes itself from the tree by sending its next hop a MACT message with the prune flag set. This process continues until a group member receives the MACT message and becomes the group leader.

6.2.6.2 Reconnecting Partitioned Trees

After a network partition, topological changes in the network can reconnect two network components. A multicast tree member from one partition will know that it has new connectivity to another partition if it receives a *Group Hello* (GRPH) message for the multicast group that contains group leader information different from its own record.

The group leader with lower IP address (GL_1) unicasts a RREQ to the other group leader (GL_2) using the node from which it received the GRPH message as the next hop. This RREQ has a *Repair flag* and GL_1's record of the multicast group sequence number. If any nodes on GL_2's tree receive the RREQ, they must forward it along a branch of the multicast tree toward GL_2. When GL_2 receives the RREQ, it takes the larger of its record of the multicast group sequence number and that indicated in the RREQ; it then increments this value by 1 and unicasts a RREP back to GL_1. This RREP also has the Repair flag set. If



any node on GL_1's multicast tree receives the RREP as it travels back to GL_1, it must forward it along the multicast tree toward GL_1. When GL_1 receives the RREP, the tree is connected. Figure 45 illustrates a tree reconnected in this manner.

6.2.7 Actions after Reboot

A node participating in the multicast tree that reboots might loose all of its multicast tree information. Upon reboot, a node should broadcast a MACT message with a set *Reboot* flag to inform neighboring nodes of the loss. When a node on the multicast tree receives the reboot MACT message, it checks whether this message came from one of its next hops on the multicast tree. If so, one of two situations exists.

If the reboot MACT came from a downstream link, the node deletes that link from its list of next hops. If the reboot MACT came from a node's upstream link, it must rebuild the tree branch.

6.2.8 Strengths and Weaknesses

Strengths: All routes are loop-free through the use of destination sequence number. Since MAODV is a on-demand routing protocol, it minimize control overhead and bandwidth utilization during periods of low mobility or minimal routing demand.

Weaknesses: Different simulation results [20, 26] shows that as node mobility increases, more and more repairs are needed to keep the tree intact. The larger network requires more repairs than the smaller network. The increased number of repairs results in a lower packet delivery ratio and a greater number of control overheads.

6.3 The Core-Assisted Mesh Protocol

The Core-Assisted Mesh Protocol (CAMP) [34, 38, 24] is a shared multicast meshes routing protocol that has richer connectivity. The main goal of using such meshes is to maintain the connectivity of multicast groups even while network routers move frequently. CAMP consists of the maintenance of multicast meshes and loop-free packet forwarding over such meshes. Within the multicast mesh of a group, packets from any source in the group are forwarded along the reverse shortest path to the source. CAMP guarantees that, within a finite time, every receiver of a multicast group has a reverse shortest path to each source of the multicast group. Multicast packets for a group are forwarded along the shortest paths from sources to receivers defined within the group's mesh. CAMP uses cores only to limit the traffic needed for a router to join a multicast group; the failure of cores does not stop packet forwarding or the process of maintaining the multicast meshes.

6.3.1 Overview of CAMP

CAMP is designed to support multicast routing in very dynamic ad-hoc networks with broadcast links. It builds and maintains a multicast mesh for information distribution within each multicast group. CAMP ensures that the shortest paths from receivers to sources (called reverse shortest paths) are part of a group's mesh. Packets are forwarded through the mesh along the paths that first reach the routers from the sources that can be defined within the mesh.

Figure 46 illustrates the differences between a multicast mesh and the corresponding shared multicast tree. A router is allowed to accept unique packets coming from any neighbor in the mesh, as opposed to

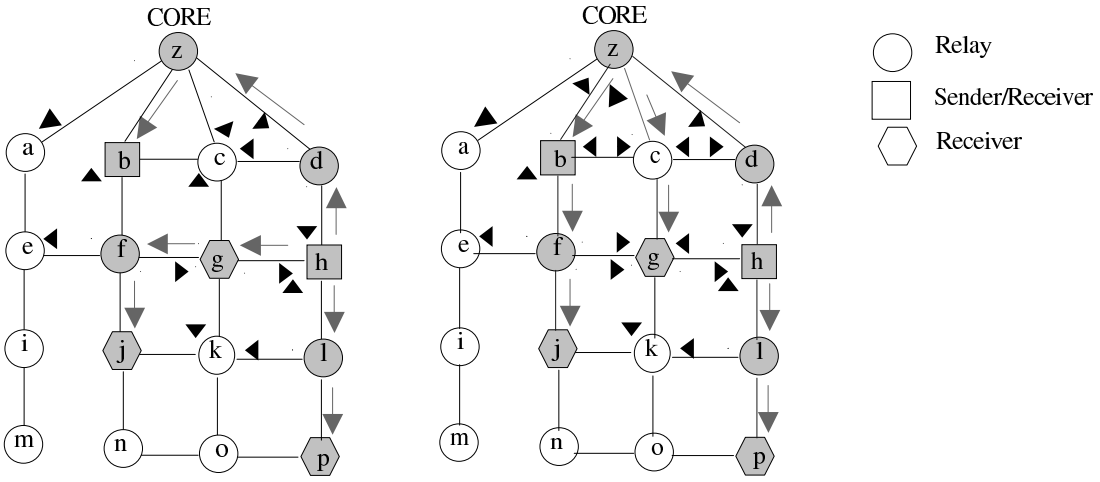


Figure 46. Traffic flow from router h in a multicast mesh (left) and in the multicast shared tree (right)

trees where a router can only take packets coming from routers with whom a tree branch has been established. Because a member router of a multicast mesh has redundant paths to any other router in the same mesh, topology changes are less likely to disrupt the flow of multicast data and to require the reconstruction of the routing structures that support packet forwarding. Routers that are members of the multicast group are shaded. The multicast mesh and tree shown in the figure include routers that have host receivers, hosts that are senders and receivers, and routers that act only as relays. Router g is the last receiver to join the multicast group, and does so in the multicast mesh through either router f or h; consequently, router c does not become a member of the mesh

All nodes in the CAMP network maintain a set of tables with membership and routing information. Moreover, all member nodes maintain a set of caches that contain previously seen data packet information and unacknowledged membership requests. CAMP classifies nodes in the network as duplex or simplex members, or non-members. Duplex members are full members of the multicast mesh, while simplex members are used to create one-way connections between sender's only nodes and the rest of the multicast mesh. Cores are used to limit the flow of Join Request packets.

CAMP consists of mesh creation and maintenance procedures. A node wishing to join a multicast mesh first consults a table to determine whether it has neighbors, which are already members of the mesh. If so, the node announces its membership via a CAMP Update. Otherwise, the node either propagates a Join Request towards one of the multicast group cores, or attempts to reach a member router by an expanding ring search of broadcasts requests. Any duplex member of the node can respond with a Join ACK, which is propagated back to the source of the request.

Periodically, a receiver node reviews its packet cache in order to determine whether it is receiving data packets from those neighbors, which are on the reverse shortest path to the source. If not, the node sends either a Heartbeat or Push Join message towards the source along the reverse shortest path. This process ensures that the mesh contains all such reverse shortest paths from all receivers to all senders. The nodes also periodically choose and refresh their selected anchors to the multicast mesh by broadcasting updates. These anchors are neighbor nodes, which are required to re-broadcast any non-duplicate data packets they receive. A node is allowed to discontinue anchoring neighbor nodes, which are not refreshing their connections. It can then leave the multicast mesh if it is not interested in the multicast session and is not required as anchor for any neighboring node.

CAMP relies on underlying unicast routing protocol, which guarantees correct distances to all destinations within finite time. Routing protocols that are based on the Bellman-Ford algorithm cannot be

used with CAMP. However, there are several recent examples of routing protocols that can be used in conjunction with CAMP [34]. With minor extensions, CAMP can also work with on-demand unicast routing protocols.

6.3.2 Information used in CAMP

Each router maintains a routing table (RT) built with the unicast routing protocol. When multicast groups need to be inserted or removed, CAMP also modified this table. CAMP assumes the existence of a beaconing protocol, usually available as a separate network service or embedded into the unicast routing protocol.

At router i, the RT made available to CAMP specifies, for each destination j, the successor (s^i_j) and the distance to the destination (D^i_j). Other than the unicast routing table, CAMP relies on the following data structures:

- CAM: table mapping cores to multicast groups.
- $CORES_g$: set of routers acting as cores to multicast group g.
- $CACHE_i$: cache of multicast data packet control information.
- MRT_i: the multicast routing table, containing the set of groups known to router i.
- AT^g_i: table containing anchor information pertaining to router i. This table is split in two subsets:
  - A^g_i: list of neighbors that have router I as their anchor for multicast group g,
  - $A2^g_i$: list of neighbors who are anchors to router i in multicast group g.

  N^g_i: router i's list of neighbors that are known to be members of the multicast group g.
- LS^g_i: list of senders that are directly attached to router i and send data traffic to multicast group g.
- LR^g_i: list of receivers directly attached to router i, who want to receive data packets from multicast group g.
- $PEND^g_i$: list of either join or simplex join requests to multicast group g originated at or forwarded by router i for whom acknowledgement is pending.
- $PENDPJ^g_i$: list of push join requests to multicast group g originated at or forwarded by router i for whom acknowledgement is pending.
- BK^g_i : list used for periodic "book-keeping" of senders and associated anchors.

When MRT_i or AT_i is updated, router i sends a multicast routing table (MRT) to all its neighbors reporting changes in its group membership and anchors per group. A router may update its MRT or AT after topology changes and messages received from its neighbors.

6.3.3 Types of cores

Multicast protocols that use cores as part of the routing structure depends on advertisement messages to propagate the list of cores currently available. With the purpose of targeting such overhead of multicast sessions on the Internet, Ohta and Crowcroft [43] propose alternate static strategies, as the use of DNS as a means to disseminate core information. Basically, by adding the CORE resource record to DNS, one could set up all cores needed for any given multicast group. As pointed by Perkins [9], the intrinsic lack of hierarchy in ad hoc networks may make DNS as it is today a hard fit for such networks. But still some name or session directory service is likely to be available.

CAMP has two different classes of cores: static and transient cores. A core defined as static has its ID obtained at system start-up from a directory service like DNS and save locally, or alternatively is defined manually by the network administrator. On the other side, transient cores assume this role when no static core is available and are expected to exist temporarily, as the name implies.

The key difference between the two classes of cores is advertisement. Since the addresses of static cores is well known, they do not need to advertise their existence, as opposed to transient ones that must advertise their addresses to the network periodically.

One important issue regarding control message overhead in ad hoc networks is the core advertisement period used by transient cores. CAMP can rely on advertisement periods with tens of seconds or even minute granularity. In the sender-initiated protocols, senders need to be known as widely and as fast as possible so that nodes all over the network can join the mesh with minimal delays. In CAMP, if a particular region of the network does not have information about any transient core, any router locally can become one and start a mesh component momentarily. This component will merge with the remainder of the mesh as the ID of other transient cores become available locally.

6.3.4 Joining and Quitting the Multicast Mesh

In CAMP, there are two different mechanisms for a router to join a multicast group mesh. If the router is attached to a receiver router, it sends a *join request* to its selected core. If instead it is attached to a host that is a source of multicast data traffic, the router will send a *simplex join request* to its preferred core. Additionally, after detecting data or receiving a list of active senders, a mesh member may request that all routers in its shortest path to a given source become mesh members with a *push join request*.

In the first mechanism, if a router joining a group has no neighbors that are already members of the multicast group, then it selects its successor to the nearest core as the relay for the join request. After the router selects a relay, it sends a join request to all its neighbors. A join request specifies the intended relay, the address of the multicast group that the sending router needs to join. If, on the other side, a router joining a group has multiple neighbors that are duplex members of the multicast group, then the router simply changes its MRT and directly announces to its neighbors that it is a new member of the multicast group using an MRU. The announcements states whether the router is a simplex or duplex member. If MRUs are sent reliably (dependind on the unicast routing protocol) the neighbor nodes acknowledge the join announcement. If MRUs are sent unreliably, the join announcement is sent periodically, so that neighbors learn about the join over time

After sending a join request, a router waits for the first acknowledgement to its request, and retransmits the request after a request-timeout. The router persists sending join requests for a number of times (e.g., four) as long as the unicast routing table indicates that there are physical paths towards any of the group cores and none of its neighbors are group members.

Any router that is duplex member of a multicast group and receives a join request for the group is free to transmit a *join acknowledgement* (ACK) to the sending router. An ACK specifies the sender of the join request and the multicast group being joined. To reduce channel traffic, the router specified as the relay of a join request can be allowed to reply first by means of a timeout mechanism after a join request is achieved.

When the origin or relay of a join request receives the first ACK to its request or the first ACK to a join request for the same multicast group, the router becomes part of the multicast group. In case of a relay, the router sends an ACK to the previous relay or origin of the join request, even if that neighbor has already sent an update stating that it is a member of the multicast group.

Cores could become hot spots if multiple non-member sources existed, and the paths followed by the packets sent by those sources could be very inefficient due to router mobility in an ad hoc network. Unlike other protocols, CAMP requires that the router attached to any source of packets for the group join the multicast mesh. To avoid the dissemination of multicast packets to routers that join a group only to allow a source-only host to send packets to the group, CAMP allows routers to belong in a multicast mesh in simplex mode, rather than as regular members.

CAMP ensures that all the reverse shortest paths between sources and receivers are part of a group's mesh by means of *heartbeat* and *push join* (PJ) messages. Periodically, every single entry in CACHE is verified. The router looks up its RT to check whether the neighbor that relayed the packet is the reverse path to the source for every cache entry. A heartbeat or a PJ is sent towards every source stored in the cache that had the number of packets coming from the reverse path under a threshold. Push join requests

may also be sent after a member gets an update message containing a list of active data traffic sources. If for any of the sources, the next hop is not known to be part of the mesh a push join request will be sent.

A router receiving a heartbeat for a given multicast group and source retransmit the heartbeat if its successor towards the source of data traffic (determined with the unicast routing protocol) is already a mesh member. When a member router receives a heartbeat and detects that its successor is not part of the multicast mesh, it sends a push join (PJ) to that neighbor router and waits for an ACK from that router.

A router that receives a PJ sends an ACK if : (a) it is the intended relay, (b) it is already a member of the group specified in the PJ, and (c) it has a path to the end point of the PJ. CAMP determines two types of push join acknowledgements- regular ACK, sent by duplex members and ACK_SIMPLEX, sent by simplex members. Given the fact that simplex mesh members do not accept packets coming from duplex members, it is important that there is no interleave of duplex and simplex routers between the initiator of a push join request and the router directly attached to the source. When acknowledgements start coming back from the source, duplex members will always send regular ACKs, and simplex members will become duplex when they receive a regular ACK. Therefore, if there is at least one duplex mesh member in the path from initiator to the source, all nodes from that duplex member all the way to the initiator must become duplex if they are not yet.

A router leaves a multicast group when it is not attached to any hosts that are members of the group and it has no neighbors for whom it is an anchor.

A router leaving a multicast group issues a *quit notification* to its neighbors, who inturn can update their MRTs. No acknowledgements are requested for quit notifications, because multicast meshes do not dictate the paths taken by multicast packets. Quit notifications are sent as part of multicast routing updates.

6.3.5 Handling Mobility and Mesh Partitioning

6.3.5.1 Link Failures

Link failures are not very critical in CAMP. When a link fails breaking the reverse shortest path to a source, the router affected by the break may not have to do anything, because the new reverse shortest path may very well be part of the mesh already; furthermore, packets keep flowing along the mesh through the remaining paths to all receivers.

Link failures have a smaller effect on CAMP, because a router joins a group with the first ACK it receives from any neighbor, and a router persists in joining while it has neighbors that are members of the mesh or its unicast routing table provides a path to a core.

6.3.5.2 Unreachable cores

CAMP reduces control traffic associated with the establishment and maintenance of multicast meshes by using multiple cores per group that routers can use as landmarks to orient their join requests. Therefore, a router can try to join a mesh orienting its unicast join requests to any of such landmarks and can redirect its join requests when topology changes. If none of the cores of a group are reachable given the unicast routing information currently available when a router needs to send a join request, this router uses an Expanded Ring Search (ERS) to reach the mesh. If the ERS can proceed and the router is not a member of the mesh, this router sends a mesh search message specifying itself as the requester. Any router receiving such a message forwards it appending its ID to the path of the message. A router that receives the mesh search message and is a mesh member replies with an acknowledgement. When the mesh search requester gets the first acknowledgement to its message, it sends a join request along the path it obtained with the acknowledgement. If it does not receive an ACK, the router retransmits its search message after a time out period. Therefore, CAMP has no single point of failure and can use as many cores as desired for a given mesh.

6.3.5.3 Keeping Meshes Connected

In CAMP, routers do not rely on a single core to join the mesh. If a multicast mesh becomes partitioned due to router mobility, CAMP still can continue the operation of all mesh components. In addition, CAMP is able to merge mesh components as long as there is physical connectivity between mesh components. To ensure that two or more mesh components with cores eventually merge, the protocol requires (a) routers to periodically check if the next hop to its selected core is part of the mesh, and (b) that all active cores in the mesh send periodical messages to each other, forcing routers along the path that are not members to join the mesh.

All members do the periodic verification of the membership of the next hop to the selected core to avoid placing of non-member routers between receivers, senders and cores. A duplex or simplex join request is re-sent towards the core, if the next hop to the core is unknown to be a mesh member. Since CAMP does not keep track of senders, periodically membership checking of the next hop to each sender is not feasible. Thus, making sure the reverse path to cores is part of the mesh provides at least a sub-optimal way for receivers to get packets generated by the senders in the group.

The messages exchanged among cores are core explicit joins (CEJ) that specify the multicast group, the intended relay of the CEJ, the intended core, and a gap flag. The flag is an information used by the receiver of a CEJ to determine whether there are non-members in a path between two cores. When the flag is kept reset all along the path between the two cores, no acknowledgement to CEJ needs to be sent back.
A router receiving the CEJ with the gap flag set to 0 forwards the CEJ to the next relay if (a) it is the specified relay and (b) it has a path to the specified core. Furthermore, if the relaying router is not a member of the mesh, it sets the gap flag to 1 in its CEJ.

A core receiving the CEJ with the gap flag set to 1 sends an ACK. The ACK is forwarded all the way back to the core that originated the CEJ; ACKs force relaying routers to join the mesh as in a PJ or a regular join. Alternatively, a router receiving the CEJ with the gap flag set to 0 forwards the CEJ to the next relay if: (a) it is the specified relay, (b) it has a path to the specified core, and (c) it is not a member of the group.

The possibility of using multiple cores in a multicast group provides fault tolerance to the protocol. But CAMP is able to merge mesh components even when no core is reachable. In, such exceptional situations, a node uses expanded ring search when it needs to join the mesh. When a search fails because no mesh member is close enough, the originator of the search request backs off, checks again for the availability of a core and if still no one is found, the originator becomes a transient core. As the existence of one or more transient cores is propagated to the network, core explicit join requests start being exchanged and different components eventually merge again.

6.3.6 Strengths and Weaknesses

Strengths:

- CAMP has much richer connectivity than trees. It does loop-free packet forwarding over meshes. CAMP guarantees that, within a finite time, every receiver of a multicast group has a reverse shortest path to each source of the multicast group. In addition, it avoids flooding data or control packet throughout the entire network.
- A router is allowed to accept unique packets coming from any neighbor in the mesh, as opposed to trees where a router can only take packets coming from routers with whom a tree branch has been established.
- Because a member router of a multicast mesh has redundant paths to any other router in the same mesh, topology changes are less likely to disrupt the flow of multicast data and to require the reconstruction of the routing structures that support packet forwarding.

Weaknesses:

- Since CAMP does not keep track of senders, checking periodically the membership of the next hop to each sender is not possible.
- Routing protocols that are based on the Bellman-Ford algorithm cannot be used with CAMP, and CAMP needs to be extended in order to work with on-demand routing protocols.
- Simulation results shows that packet loss is slightly higher than their ODMRP counterparts.

6.4 On - Demand Multicast Routing Protocol (ODMRP)

On-Demand Multicast Routing Protocol (ODMRP) is a multicast routing protocol developed by the Wireless Adaptive Mobility (WAM) Laboratory [54, 56] at University of California, Los Angeles, for ad hoc networks with mobile hosts. ODMRP is a mesh-based, rather than a conventional tree-based, multicast scheme and uses a forwarding group concept; only a subset of nodes forwards the multicast packets via scoped flooding. It applies on-demand procedures to dynamically build routes and maintain multicast group membership. ODMRP is well suited for ad hoc wireless networks with mobile hosts where bandwidth is limited, topology changes frequently and rapidly, and power is constrained. By maintaining and using a mesh instead of a tree, the drawbacks of multicast trees in mobile wireless networks (e.g., intermittent connectivity, traffic concentration, frequent tree reconfiguration, non-shortest path in a shared tree, etc.) are avoided. A soft-state approach is taken to maintain multicast group members. No explicit control message is required to leave the group. The reduction of channel/storage overhead and the relaxed connectivity may make ODMRP more attractive in mobile wireless networks.

6.4.1 Multicast Route and Mesh Creation

ODMRP established and maintains group membership and multicast routes by the source on demand. Similar to on-demand unicast routing protocols, a query phase and a reply phase comprise the protocol (refer to Figure 47). While a multicast source has packets to send but no route and group membership is known, it floods a member advertising packet with data payload piggybacked. This packet, called JOIN QUERY, is periodically broadcasted to the entire network to refresh the membership information and updates the routes as follows. When a node receives a non-duplicate JOIN QUERY, it stores the upstream node ID (i.e., backward learning) into the routing table and rebroadcasts the packet. When a JOIN QUERY packet reaches the multicast receiver, it creates and broadcasts a JOIN REPLY to its neighbors. When a node receives a JOIN REPLY, it checks if the next node ID of one of the entries matches its own ID. If it does, the node realizes that it is on the path to the source and thus is part of the forwarding group.

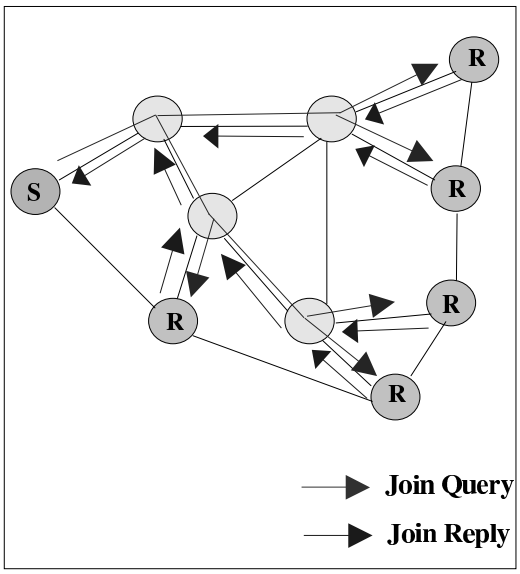


Figure 47. On-demand procedure for membership setup and maintenance

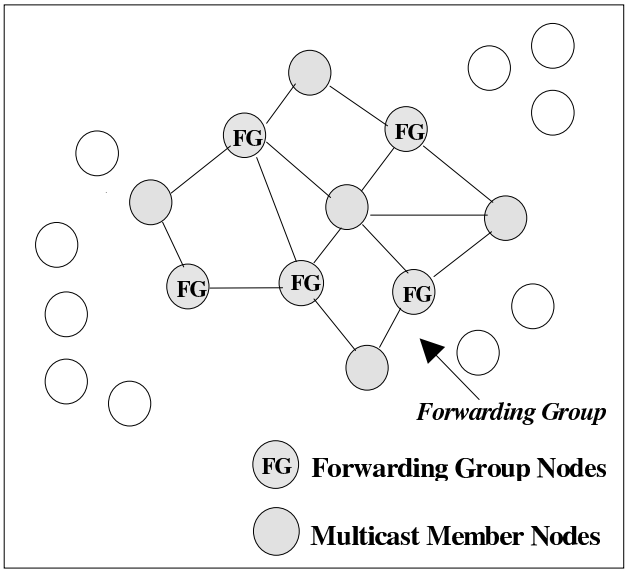


Figure 48. The forwarding group concept

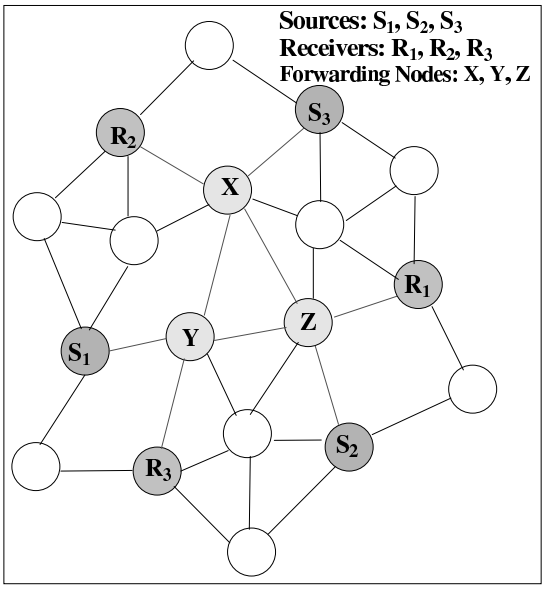


Figure 49. Why a mesh.

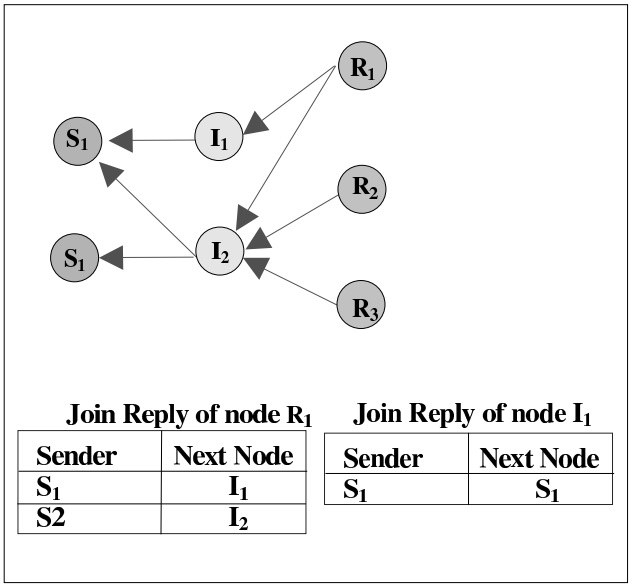


Figure 50. An example of a JOIN REPLY forwarding.

It then sets the FG_FLAG (Forwarding Group Flag) and broadcasts its own JOIN REPLY built upon matched entries. This way, the JOIN REPLY is propagated by each forward group member until it reaches the multicast source via the shortest path. This process constructs (or updates) the routes from sources to receivers and builds a mesh of nodes, the forwarding group.

Figure 48 illustrates the forwarding group concept. The forwarding group is a set of nodes, which is in charge of forwarding multicast packets. It supports shortest paths between any member pairs. All nodes can forward multicast data packets if they are inside the "bubble" (multicast members and forwarding group nodes). If a multicast receiver is on the path between a multicast source and another receiver, it also can be a forwarding group node. Compared with trees, the mesh provides richer connectivity among multicast members. Node displacements and channel fading overcome with the help of route redundancy among forwarding group. Therefore, unlike trees, frequent reconfigurations are not required.

Figure 49 shows the robustness of a mesh configuration. Three sources (S_1, S_2, and S_3) send multicast data packets to three receivers (R_1, R_2, and R_3) via three forwarding group nodes (X, Y, and Z). Suppose the route from S_1 to R_1 is < S_1-Y-Z-R_1 >. In a tree configuration, if a link between nodes Y and Z breaks or fails, R_1 cannot receive any packets from S_1 until the tree is reconfigured. On the other hand, ODMRP already has a redundant route in < S_1-Y-X-Z-R_1 > to deliver packets without going through the broken link between nodes Y and Z.

6.4.2 Example

Let us consider the above Figure 50 as an example of JOIN REPLY forwarding process. Nodes S_1 and S_2 are multicast sources, and nodes R_1, R_2, and R_3 are multicast receivers. Nodes R_2 and R_3 send their JOIN REPLIES to both S_1 and S_2 via I_2. R_1 sends its JOIN REPLY to S_1 via I_1 and to S_2 via I_2. When receivers send their JOIN REPLIES to next hop nodes, an intermediate node I_1 sets the FG_FLAG and builds its own JOIN REPLY since there is a next node ID entry in the JOIN REPLY received from R_1 that matches its ID. Note that the JOIN REPLY built by I_1 has an entry for sender S_1 but not for S_2 because the next node ID for S_2 in the received JOIN REPLY is not I_1. In the meanwhile, node I_2 sets the FG_FLAG, constructs its own JOIN REPLY and sends it to its neighbors. Note that I_2 broadcasts the JOIN REPLY only once even though it receives two JOIN REPLIES from the receivers because the second table arrival

carries no new source information. Channel overhead is thus reduced dramatically in cases where numerous multicast receivers share the same links to the source.

6.4.3 Reliability

In establishing and refreshing multicast routes and forwarding groups the reliable transmission of JOIN REPLIES plays an important role. Hence, if JOIN REPLIES are not properly delivered, effective multicast routing cannot be achieved by ODMRP. The IEEE 802.11 MAC (Medium Access Control) protocol [8], performs reliable transmission by retransmitting the packet if no acknowledgment is received. However, if the packet is broadcasted, no acknowledgments or retransmissions are sent. In ODMRP, the transmissions of JOIN REPLIES are often broadcasted to more than one upstream neighbor since we are handling multiple sources. In such cases, the hop-by-hop verification of JOIN REPLY delivery and the retransmission cannot be handled by the MAC layer. It must be done indirectly by ODMRP. Another option for reliable delivery is to subdivide the JOIN REPLY into separate sub-tables, one for each distinct next node. In Figure 50 for example, the JOIN REPLY at node R_1 is split into two Join Replies, one for neighbor I_1 and the other for neighbor I_2. These Join Replies are separately unicasted using a reliable MAC protocol such as IEEE 802.11 or MACAW.

Figure 51 illustrates the scheme that was used in [35]. When node B transmits a packet to node C after receiving a packet from node A, node A can hear the transmission of node B if it is within B's radio propagation range. Hence, the packet transmission by node B to node C is used as a "passive acknowledgement" to node A. We can utilize this passive acknowledgment to verify the delivery of a JOIN REPLY. Note that the source itself must send an active acknowledgment to the previous hop since it does not have any next hop to send a JOIN REPLY to unless it is also a forwarding group node for other sources.

A node may not hear the passive acknowledgments of its upstream neighbor because of conflicts due to the hidden terminal problem. It will also not hear the passive acknowledgment if the upstream neighbor has moved away. In either case, when no acknowledgment is received within the timeout interval, the node retransmits the message. If packet delivery cannot be verified after an appropriate number of

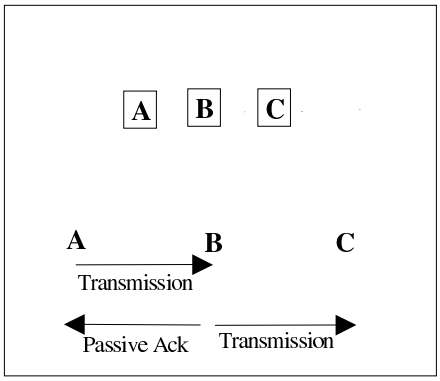


Figure 51. Passive acknowledgements.

retransmissions, the node considers the route to be invalidated. To find an alternate route "on the spot" the node thus broadcasts a message to its neighbors specifying that the next hop to a set of sources cannot be reached. Upon receiving this packet, each neighbor builds and unicasts the Join Reply to its next hop if it has a route to the multicast sources. If no route is known, it simply broadcasts the packet specifying the next hop is not available. In both cases, the node sets its FG_FLAG. The FG_FLAG setting of every neighbor may create excessive redundancy, but most of these settings will expire because only necessary forwarding group nodes will be refreshed in the next JOIN REPLY propagation phase.

6.4.4 Data Forwarding

After this group establishment and route construction process, a source can multicast packets to receivers via selected routes and forwarding groups. When receiving the multicast data packet, a node forwards it only when it is not a duplicate and the setting of the FG_FLAG for the multicast group has not expired. This procedure minimizes the traffic overhead and prevents sending packets through stale routes.

6.4.5 Soft State

In ODMRP, no explicit control packets need to be sent to leave the group. If a multicast source wants to leave the group, it simply stops sending any JOIN QUERY packets since it does not have any multicast data to send to the group. If a receiver no longer wants to receive from a particular multicast group, it does not send the JOIN REPLY for that group. Nodes in the forwarding group are demoted to non-forwarding nodes if not refreshed (no JOIN REPLIES received) before they timeout.

6.4.6 Selection of Timer Values

Timer values for route refresh interval and forwarding group timeout interval can have impacts on ODMRP performance. The selection of these soft state timers should be adaptive to network environment (e.g., traffic type, traffic load, mobility pattern, mobility speed, channel capacity, etc.). When small route refresh interval values are used, fresh route and membership information can be obtained frequently at the expense of producing more packets and causing network congestion. On the other hand, when large route refresh values are selected, even though less control traffic will be generated, nodes may not know up-to-date route and multicast membership. Thus in highly mobile networks, using large route refresh interval values can yield poor protocol performance. The forwarding group timeout interval should also be carefully selected. In networks with heavy traffic load, small values should be used so that unnecessary nodes can timeout quickly and not create excessive redundancy. In situations with high mobility, however, large values should be chosen so that more alternative paths can be provided. It is important to note that the forwarding group timeout value must be larger (e.g., 3 to 5 times) than the value of route refresh interval.

6.4.7 Unicast Capability

One of the major strengths of ODMRP is its unicast routing capability. Not only can ODMRP coexist with any unicast routing protocol, it can also operate very efficiently as an unicast routing protocol. Thus, a network equipped with ODMRP does not require a separate unicast protocol. Other ad hoc multicast routing protocols such as AMRoute, CAMP, RBM, and LAM must be run on top of a unicast routing protocol. CAMP, RBM, and LAM in particular, only work with certain underlying unicast protocols.

6.4.8 Data Structures

Nodes running ODMRP are required to maintain the following tables.

6.4.8.1 Routing Table

A routing table is created on demand and is maintained by each node. An entry is inserted or updated when a non-duplicate JOIN QUERY is received. The node stores the destination (i.e., the source of the JOIN QUERY) and the next hop to the destination (i.e., the last node that propagated the JOIN QUERY). The routing table provides the next hop information when transmitting JOIN REPLIES.

6.4.8.2 Forwarding Group Table

When a node is a forwarding group node of the multicast group, it maintains the group information in the forwarding group table. The multicast group ID and the time when the node was last refreshed are recorded.

6.4.8.3 Message Cache

The message cache is maintained by each node to detect duplicates. When a node receives a new JOIN QUERY or data, it stores the source address and the unique identifier of the packet. Note that entries in the message cache need not be maintained permanently. Schemes such as LRU (Least Recently Used) or FIFO (First In First Out) can be employed to expire and remove old entries and prevent the size of the message cache to be extensive.

6.4.9 Mobility Prediction

6.4.9.1 Adapting the Refresh Interval via Mobility Prediction

To build and refresh routes ODMRP requires periodic flooding of JOIN QUERY. Because of bandwidth constraints excessive flooding is not desirable in ad hoc networks. In addition, flooding often causes congestion, contention, and collisions. Finding the optimal flooding interval is critical in ODMRP performance. In highly mobile networks where nodes are equipped with GPS [11] (e.g., tactical networks with tanks, ships, aircrafts, etc.), we can efficiently adapt the REFRESH_INTERVAL to mobility patterns and speeds by utilizing the location and movement information. The location and movement information are used to predict the duration of time routes will remain valid. With the predicted time of route disconnection, JOIN QUERIES are only flooded when route breaks of ongoing data sessions are imminent.

A free space propagation model is assumed, where the received signal strength solely depends on its distance to the transmitter. It is also assume that all nodes in the network have their clock synchronized (e.g., by using the NTP (Network Time Protocol) or the GPS clock itself). Therefore, if the motion parameters of two neighbors (e.g., speed, direction, radio propagation range, etc.) are known, we can determine the duration of time these two nodes will remain connected. Assume two nodes i and j are within the transmission range r of each other. Let (x_i, y_i) be the coordinate of mobile host i and (x_j, y_j) be that of mobile host j. Also let v_i and v_j be the speeds, and θ_i and θ_j $(0 \leq \theta_i, \theta_j < 2\pi)$ be the moving directions of nodes i and j, respectively. Then, the duration of time that the link between two nodes will stay connected, D_t, is given by:

$$D_t = \frac{-(ab + cd) + \sqrt{((a^2 + c^2)r^2 - (ad - bc)^2)}}{a^2 + c^2}$$

where

$a = v_i\cos\theta_i - v_j\cos\theta_j$, $b = x_i - x_j$, $c = v_i\sin\theta_i - v_j\sin\theta_j$, and $d = y_i - y_j$.

Note that when $v_i = v_j$ and $\theta_i = \theta_j$, D_t is set to infinity without applying the above equation.

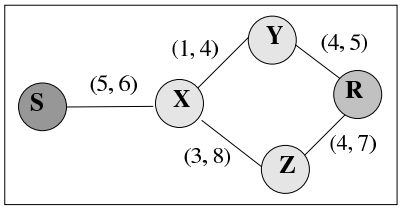
Extra fields must be added into JOIN QUERY and JOIN REPLY packets to utilize the information obtained from the prediction. When a source sends a JOIN QUERY, it appends its location, speed, and direction. Since the source does not have any previous hop node it sets the MIN_LET (Minimum Link Expiration Time) field to the MAX_LET_VALUE. The next hop neighbor, upon receiving a JOIN QUERY, predicts the link expiration time between itself and the previous hop using the above equation. The minimum between this value and the MIN_LET indicated by the JOIN QUERY is included in the packet. The rationale is that as soon as a single link on a path is disconnected, the entire path is invalidated. The node also overwrites the location and mobility information field written by the previous node with its own information. When a multicast member receives the JOIN QUERY, it calculates the predicted LET of the last link of the path. The minimum between the last link expiration time and the MIN_LET value specified in the JOIN QUERY is the RET (Route Expiration Time). This RET value is enclosed in the JOIN REPLY and broadcasted. If a forwarding group node receives multiple JOIN REPLIES with different RET values (i.e., lies on paths from the same source to multiple receivers), it selects the minimum RET among them and sends its own JOIN REPLY with the chosen RET value

attached. When the source receives JOIN REPLIES, it selects the minimum RET among all the JOIN REPLIES received. Then the source can build new routes by flooding a JOIN QUERY before the minimum RET approaches (i.e., a route breaks).

When choosing the refresh interval, in addition to the estimated RET value, other factors need to be considered. Routes will expire quickly and often, if the node mobility rate is high and the topology changes frequently. The source may propagate JOIN QUERY excessively and this excessive flooding can cause collisions and congestion, and clogs the network with control packets. Thus, the MIN_REFRESH_INTERVAL should be enforced to avoid control message overflow. On the other hand, routes will hardly expire and the source will rarely send JOIN QUERY if nodes are stationary or move slowly and link connectivity remains unchanged for a long duration of time. A few problems arise in this situation. First, if a node in the route suddenly changes its movement direction or speed, the predicted RET value becomes obsolete and routes will not be reconstructed in time. Second, when a non-member node which is located remotely to multicast members wants to join the group, it cannot inform the new membership or receive data until a JOIN QUERY is received. Hence, the MAX_REFRESH_INTERVAL should be set. The selection of the MIN_REFRESH_INTERVAL and the MAX_REFRESH_INTERVAL values should be adaptive to network situations (among others, traffic type, traffic load, mobility pattern, mobility speed, channel capacity).

6.4.9.2 Route Selection Criteria

In ODMRP, a multicast receiver selects routes based on the minimum delay (i.e., routes taken by the first JOIN QUERY received). A different route selection method is applied when we use the mobility prediction. The idea is inspired by the Associativity-Based Routing (ABR) protocol which chooses associatively stable routes. In our algorithm, instead of using the minimum delay path, we can choose a route that is the most stable (i.e., the one that will remain connected for the longest duration of time). To select a route, a multicast receiver must wait for an appropriate amount of time after receiving the first JOIN QUERY so that all possible routes and their route qualities will be known. The receiver then chooses the most stable route and broadcasts a JOIN REPLY. Route breaks will occur less often and the number of JOIN QUERY propagation will reduce because stable routes are used. An example showing the difference between two route selection algorithms is presented in Figure 52. Two routes are available from the source S to the receiver R. Route 1 has the path of < S-X-Y-R > and route 2 has the path of < S-X-Z-R >. If the minimum delay is used as the route selection metric, the receiver node R selects route 1. Route 1 has a delay of (5+1+4 = 10) while route 2 has a delay of 11 (5+3+4 = 11). Since the JOIN QUERY that takes route 1 reaches the receiver first, node R chooses route 1. If the stable route is selected instead, the receiver chooses route 2. The route expiration time of route 1 is 4 (min (6, 4, 5) = 4) while that of route 2 is 6 (min (6, 8, 7) = 6). The receiver selects the route with the maximum RET, and hence route 2 is selected.



	Route 1	Route 2
Path	S-X-Y-R	S-X-Z-R
Delay	10	12
RET	4	6

(i, j) Link with delay i and link expiration time j

Figure 52. Route selection example

6.4.9.3 Alternative Method of Prediction

Since GPS may not work properly in certain situations (e.g., indoor, fading, etc.), we may not always be able to accurately predict the link expiration time for a particular link. However, there is an alternative method to predict the LET. This method is based on a more realistic propagation model. Basically,

transmission power samples are measured periodically from packets received from a mobile's neighbor. From this information it is possible to compute the rate of change for a particular neighbor's transmission power level. Therefore, the time when the transmission power level will drop below the acceptable value (i.e., hysteresis region) can be predicted. This option requires further investigation.

6.4.10 Strengths and Weakness

Strengths: The following properties of ODMRP highlight its strengths.

- Simplicity
- Low channel and storage overhead
- Usage of up-to-date shortest routes
- Reliable construction of routes and forwarding group
- Robustness to host mobility
- Maintenance and exploitation of multiple redundant paths
- Exploitation of the broadcast nature of wireless environments
- Unicast routing capability

In addition, providing multiple paths by formation of mesh configuration makes the protocol robust to mobility. Alternate route enables data delivery in the face of mobility and link breaks while the primary route is being reconstructed. The protocol does not yield excessive channel overhead in highly mobile networks because no control packets are triggered by the link breaks. Another advantage of this protocol is its unicast capability. Finally, the reduction of channel/storage overhead and the richer connectivity make ODMRP more attractive in mobile wireless networks.

Weaknesses: ODMRP suffers from excessive flooding when there are a large number of multicast senders.

7 Multicast Comparison

This chapter summarizes key characteristics and properties of the previously described multicast protocols. Table 13 presents a comparison of MAODV, ODMRP, AMRoute and CAMP.

The discussion here is based on Table 13. MAODV shares certain salient characteristics with ODMRP. In particular, they both discover routes only when data packets lack a route to a multicast destination. Route discovery in either protocol is based on request and reply cycles, and multicast route information is stored in all intermediate nodes along the multicast path. However, there are several important differences in the dynamics of these two protocols, which may give rise to significant performance differentials. The most notable of these is that the MAODV uses a shared bi-directional multicast tree for forwarding data packets while ODMRP maintains a mesh topology rooted from each source. In MAODV, if there is a

Protocols	MAODV	ODMRP	AMRoute	CAMP
Configuration	Tree	Mesh	Tree	Mesh
Loop-Free	Yes	Yes	No	Yes
Dependency on Unicast Protocol	No	No	Yes	Yes
Periodic Message	Yes	Yes	Yes	Yes
Control Packet Flood	Yes	Yes	Yes	No
Sleep Period Operation	No	No	No	No
Security	No	No	No	No

Table 13. Comparisons of the characteristics of multicast routing protocols.

partition in a multicast group due to link failure, there are no other alterative paths between source and destination. On the other hand, ODMRP provides alternative paths and a link failure need not trigger the recomputation of the mesh from sources to the receivers. In addition, ODMRP broadcasts the reply back to the source while MAODV unicasts the reply back to the source. Also, MAODV does not activate a multicast route immediately while ODMRP does. Furthermore, MAODV sends control messages to repair broken links and to manage network partition while ODMRP uses a soft state approach, and wait for broken links timeout. In MAODV, a multicast group leader maintains up to date multicast tree information, while ODMRP source nodes periodically send request messages in order to refresh the multicast mesh. Finally, MAODV uses an expanding ring broadcast control packets, while ODMRP sends all broadcast messages through the network.

AMRoute and MAODV are both tree based protocol. AMRoute creates a bi-directional shared multicast tree using unicast tunnels to provide connections between multicast group members. Each group has at least one core that is responsible for member and tree maintenance. On the other hand, in MAODV, as nodes join the multicast group, a bi-directional multicast tree composed of group members and nodes used to connect them is created. Each multicast group has associated with it a group leader. The multicast group leader maintains the multicast group sequence number.

AMRoute suffers from temporary loops and creates non-optimal trees when mobility is present. All other multicast protocols discussed in this paper are loop-free.

AMRoute and CAMP do require an underlying unicast protocol to operate, but ODMRP and MAODV do not. While AMRoute can work with any protocol, the designers of CAMP specifically state that it can operate only with certain unicast protocols [34]. The unique property of ODMRP is its unicast capability. Not only can ODMRP coexist with any unicast routing protocol, it can also operate well as unicast routing protocol.

To date, CAMP is the only multicast routing protocol not based on trees that avoids flooding of data or control packets to establish the routing structure for a group. Although CAMP and ODMRP use a different mesh approach, they share some common features. The idea of anchors is present in both - when a router reads a member table in and finds out it has to set the forwarding flag, the router is becoming an anchor for the neighbor sending the member table. Rather than using reverse path forwarding, both protocols rely on packet caching to avoid loops. The major difference is the sender-initiated approach used in ODMRP, which requires control packets to be flooded in the network. AMRoute and MAODV also use control packet flooding approach. All four multicast protocols described in this paper use periodic message advertisement. None of the multicast protocols described in this report support security and sleep mode.

8 Multicast Performance Comparison

As I mentioned earlier in this report, when comparing two ad hoc networking protocols there are certain metrics, which have been established to be most useful in deciding which ones 'perform' better. These include: measurements of end-to-end data throughput and delay, route acquisition time, percentage out of order delivery, and efficiency (based on bandwidth overhead). Certain simulators provide the ability to measure some of these metrics better than others. To date there are not enough simulation results available on multicast performance comparison study to compare various parameters. Only very few groups performed simulations of ad hoc wireless multicast protocols in order to do a performance comparison. For the most part the environment parameters and metrics used by these groups were very much inconsistent.

Lee et al. [53] and Perkins & Royer [9,20] have been performed their simulation using GloMoSim simulation package. Lee et al.'s research then consisted of a performance comparison of AMRoute, ODMRP, AMRIS and CAMP. Perkins and Royer compile performance data on MAODV [22]. The remaining paper, Garcia et al. used the C++ Protocol Toolkit (CPT), from Rooftop Communication [34] and focused on the performance of CAMP and ODMRP.

Lee et al.'s simulation results have shown that in a mobile scenario, mesh based protocols outperformed tree-based protocols. The availability of alternate routes provided robustness to mobility. AMRoute performed well under no mobility, but it suffers from loops and inefficient trees even for low mobility. CAMP showed better performance when compared to tree protocols, but with mobility, excessive control overhead caused congestion and collisions that resulted in performance degradation. ODMRP was very

effective and efficient in most of the simulation scenarios. However, the protocol showed a trend of rapidly increasing overhead as the number of senders increased.

Madruga et al [24] have also shown in their simulation experiments that mesh-based multicast protocols outperform tree-based multicast protocol in dynamic networks. Their simulation experiments also show that the receiver-initiated approach used for mesh joining in CAMP performs and scales better than the sender-initiated approach of ODMRP. The protocol performance in the experiments illustrates that meshes can be used effectively as multicast routing structures without the need for flooding of the control packets.

The simulation results presented by Cheng in his M. Eng. thesis [26] show that ODMRP outperforms MAODV in terms of packet delivery. However, ODMRP suffers from scalability issues as the number of senders increase. They demonstrated the improvement of packet delivery for MAODV, at high levels of mobility by using immediate route activation. They used self-pruning in MAODV and ODMRP to decrease the control overhead for packets that are flooded through the network.

Perkins et al.'s [9] simulation results illustrate that in MAODV, for zero mobility, the number of received data packets is between 95% and 100%. Because of the possibility of collisions, packet delivery of 100% is not achieved for zero mobility. As the speed of node movement increases, the packet delivery ratio decreases as it becomes more and more difficult to maintain the multicast tree links. For slow node mobility, however, the protocol performs well, achieving between 87% and 97% packet delivery. As node movement increases, the smaller network outperforms the larger network by an increasingly greater margin.

Again, as node mobility increases, the number of control messages also increases because the need to repair multicast tree branches is more frequent to keep the tree intact. The larger network requires more repairs than the smaller network. The increased number of repairs results in a lower packet delivery ratio and a greater number of control overheads.

From the above discussion, it clear that mesh protocols performed significantly better than the tree protocols in the presence of high mobility. In trees, when routes are invalidated due to node movements, the packets must be buffered or dropped until the tree is reconfigured. On the other hand, redundant routes in the mesh provide alternate routes for data delivery in the face of mobility and link breaks. Data packets can still reach the destinations while the primary route is being reconstructed. However, mesh based protocols also still have many open issues including scalability, non-uniform traffic, multiple multicast groups, security, QoS etc. Further research must be done to address all remaining open issues prior to determining the superiority of one protocol over the other.

9 Challenges of Ad Hoc Wireless Networks

Ad hoc Networking is a rather hot concept in computer communications. This means that there is much research going on and many issues that remain to be solved.

9.1 Problems remain to be solved at different layers of the protocol stack

Link layer: Medium access control (MAC) protocols that exploit the capabilities of the physical layer need to be developed. For instance, the physical layer may be capable of performing power control, using multiple modulations or coding schemes, or of providing channel-condition-related information to the upper layers. MAC protocols capable of utilizing these capabilities need to be developed to optimize performance.

Due to the error-prone nature of wireless links, it is useful to employ suitable link-level retransmission schemes. One important factor affecting this choice is the nature of packets being retransmitted. For instance, if the packets belong to a delay-sensitive flow that can tolerate some packet losses, a highly persistent retransmission scheme may not be useful. On the other hand, if the packets belong to a TCP connection, perhaps such a link-level retransmission scheme would help. In general, the choice of retransmission scheme would also depend on factors such as error distribution and delays on the wireless channel.

Fair sharing of bandwidth is another issue of interest. Some MAC protocols make it possible for a node to utilize a disproportionate fraction of channel bandwidth, while other nodes are not able to get their fair share. Techniques to improve the fairness of MAC protocols are being investigated at present.

Network layer: Since the nodes in a mobile ad hoc network may move often, it is necessary to use an efficient routing protocol. There has been significant activity on developing routing protocols for mobile ad hoc networks. These routing protocols may be broadly divided into two categories: proactive and reactive. Proactive protocols tend to maintain routes between all node pairs, regardless of whether or not those routes are actually being used. The traditional link state and distance vector routing protocols belong in this category. In contrast, reactive protocols tend to explicitly establish routes on demand and maintain only the routes that are likely to be used in the near future. The reactive approach has been shown to often perform better than the proactive approaches, since the reactive approach tends to reduce unnecessary route maintenance traffic. Ideally, a routing protocol should be able to adapt to the particular environment, by incorporating both proactive and reactive behaviors. Design of such routing protocols is a topic of ongoing research.

In addition to unicast routing, there has been much activity on multicast routing protocols for ad hoc networks. Similar to unicast routing, these protocols need to be able to deal with topology changes due to node mobility.

Transport layer: In mobile ad hoc networks, packet losses can occur due to congestion, transmission errors, or host mobility. Existing transport protocols, particularly TCP, deal quite well with congestion-related packet losses. However, new or modified transport protocols may have to be developed to take into account other types of packet losses.

Application layer: Ultimately, the success of ad hoc networking will depend on whether interesting applications exist for such networks. Many of the current research activities are focused on other issues at the lower layers, as described above. There is a need for further work on applications for ad hoc networks to help the current efforts bear fruit.

9.2 Several other issues of interest span various layers of the protocol stack

- Scalability - how large can an ad hoc network grow?
- Quality of Service - can bandwidth or delay-constrained applications operate well?
- Client-server model shift - what happens when a client cannot count on traditional methods to locate a suitable server?
- Security - since ad hoc is reliant on wireless communication, it is important to incorporate mechanisms to protect data transmitted on the wireless link and forwarded through intermediate nodes. Is there a good way to protect against attacks from malicious ad hoc nodes?
- Interoperation with the Internet - how can an ad doc network take advantage of evanescent or dynamically changing points of connection to the Internet?
- Power Control - Conservation of power and power-aware routing must be taken into consideration. Routing may be desirable on more "long-lived" routes. How can battery life be maximized?
- Since the topology of the network is constantly changing, the primary challenge is routing with frequent link failure and disconnections. The distribution of up-to-date information can easily saturate a network; late arrival of information can drive a network into instability.
- Multicast routing will be a challenge because the multicast tree is no longer static. i.e its topology is subject to change over time.
- Up till now most evaluations have been done with a single multicast group. Evaluation needs to be done to look at the impact and interaction between multiple multicast groups in the network
- Providing different quality of service levels in a constantly changing environment will be a challenge.
- In accommodating packets with preemptive priorities, the network may not be able to preserve the QoS guarantee for ordinary flows.

- The development of QoS routing policies, algorithms, and protocols for handling user data with multiple priorities is also an open area.
- The need for location-aided routing. This may be done by using positioning information to define regions so that cluster routing may be accomplished.
- Adaptive Mobile Applications - Mobile applications should be able to dynamically manage resources, such as memory, processing time, disk accesses, to help take advantage of the dynamic nature of the bandwidth in ad hoc networks.
- Fault Tolerance in Mobile Computing - Wireless communication is subject to many types of problems due to interference and poor signals. Fault tolerance can be built into the hardware and software to help deal with errors appearing during transmission.
- Nomadic Collaboration Framework - A more specific framework may be developed to facilitate collaboration. This may include a more specialized protocol, communications interface, and data management.
- Integration with Broadband ATM - By combining ad hoc technology and ATM technology, it would be possible to deliver efficient/high data rate connections to mobile computer users.
- Cross-layer interaction**:** Traditionally, different layers of the protocol stack have been designed to minimize interaction between the layers. However, in the context of wireless networks, it appears that much is to be gained by increasing this interaction. For instance, the link layer may be able to improve performance by knowing more about channel conditions. Similarly, the transport layer may benefit by knowing whether a packet was lost due to route failure or congestion. Alternatively, depending on the protocol employed, lower layers may gain from having some information on the nature of the data from the higher layer. Current research is investigating such information exchange between layers of the protocol stack.

9.2.1 Some of the unanswered questions will be briefly considered here.

Scalability: Because of many interacting factors, no one knows just how large an ad hoc network can grow. It is safe to say that such a network cannot grow to the size of the Internet, which has tens of millions of IP addressable devices. So far, most simulations have been done with only 50 or 100 nodes, and even these small populations have exhibited fairly poor performance using the routing techniques available today. Royer et. al**.** [9] run simulations of 10,000 nodes, but the simulation environment definitely interferes with such experiments. One simulation can easily take over a gigabyte of memory and equal amounts of disc storage depending on which parameters are being measured.

With on-demand protocols, one can deploy larger populations of mobile nodes in the ad hoc network by accepting worse performance for route acquisition latency. Looking at it the other way around, demanding very short latencies for route acquisition can place heavy (or impossible) constraints on network size. Starting from a network without structure or valid route cache entries, the minimum route acquisition latency that allows full connectivity is the product of the maximum diameter of the network multiplied by the minimum node traversal time for route requests. At any point, the average route acquisition latency can be much better or much worse than this depending on average dissemination of valid route information and network congestion.

Lastly, I should mention that ad hoc networking ideas exhibiting sufficient scalability while still allowing subnet aggregation may find natural application in the Internet at large. This would help the problem of route thrashing in the Internet, which is partially caused by routes that change too fast to be faithfully managed by the routing protocols in use. Conversely, we might find that continued scalability to large node populations might require some advanced routing techniques from wide-area routing protocols. However, I believe that such techniques are rendered unnecessary by the on-demand nature of several leading ad hoc network protocols. It seems less likely that the Internet will ever borrow this idea of on-demand route acquisition

Quality of Service: Many of the candidate protocols operate to establish end-to-end connectivity between network applications by finding a communication path between the endpoints without regard to the quality of the physical links between intermediate points. Thus, for these approaches it would be difficult to know whether any particular application bandwidth or delay requirements can be supported by the communications path.

For many traditional audio and video applications, specifically two-way voice communications (i.e., telephony), it is quite likely that the chosen communication path between the endpoints will have to meet additional constraints. The application will have to supply its constraint parameters (i.e., its QoS parameters) to the routing layer so that a suitable path can be found.

The path can be found in several ways. For instance, with a protocol that maintains full link-state information dynamically for every link in the ad hoc network, the well-known Dijkstra algorithm can be applied, taking into account whether each link under consideration meets the application's requirements. Alternatively, almost any on-demand protocol can be equipped with extensions describing the requirements so that only appropriate paths are returned during route discovery. This is the approach taken by AODV [9,22].

All approaches are vulnerable to dynamic link quality variations between neighboring nodes. Thus, a route between two end points that initially meets the application's QoS constraints may soon fail to meet them. Methods for detecting and reporting these dynamic failure conditions need to be investigated and integrated into existing protocols. For instance, a new ICMP message (perhaps named ICMP QOS_LOST) might be defined to inform one or both endpoints that a new route discovery operation should be initiated.

Is the Client-Server Model Viable: For many years, network applications have been designed to offer their users a *client-server* model. Typically, this means that a network client is configured to use a server (often, a remote server) as its partner (or sometimes as its adult supervisor) for network transactions. More advanced programs allow the network client to be automatically configured with the identity of an appropriate server (e.g., the DNS name or perhaps the IP address). Most recently, standards have been produced to enable clients to perform dynamic discovery algorithms that carry out the automatic configuration.

However, when the network is itself dynamic, and when its structure is no longer defined by techniques for collecting IP addresses into subnets, existing techniques for static and automatic configuration break down almost completely. Static configuration is out of the picture because an ad hoc network is characterized by its dynamic nature. Automatic configuration often depends on client transactions with a configuration server such as DHCP, and DHCP typically manages client parameters based on the subnet to which the client is connected. This make today's deployed systems for automatic configuration inappropriate for ad hoc networks.

Where do services reside? That question is often not easy to answer, especially when the answer changes over time. Recent proposals have suggested integrating service discovery with the general route discovery mechanism. This can be done by specifying that only a particular kind of service (or perhaps only an application at a particular TCP or UDP port) be allowed to answer the broadcast request. Such an approach may have three deficiencies.

- Inserting application service discovery into a network layer protocol may be a violation of modular network protocol design.
- It may not be possible for the client to specify the exact service it desires in a way that can be naturally included in a network layer request.
- It may be difficult to make any determination about authorization at the network layer.

Other possibilities include the use of well-known multicast addresses for services and the service location protocol. The former approach may be more appropriate for very basic services such as DNS and DHCP. However, the question of administering and maintaining such servers in an ad hoc network is completely

open, as far as I know. More esoteric services such as certificate authorities and SNMP management agents are completely off the map.

One further idea: Whenever some of these more basic questions are answered, a next step might be to maintain a *grid* of well-known services, with new candidate servers dynamically appear in order to preserve a minimum number of hops to the closest service (e.g., of the above-mentioned types).

Connecting to the Internet: It is now possible to say a few things about how an ad hoc network can establish overall connectivity to the global Internet, but many (or perhaps most) of the problems are very poorly understood.

If any node in an ad hoc network also has connectivity to the global Internet, it is advantageous for that node to offer Internet connectivity to the other nodes. This can be done in several ways. We might think of having the Internet "gateway" advertise itself as a default router, which will have the desired effect if the other nodes in the ad hoc network can consider themselves connected to the default router by way of a multihop path through other ad hoc network nodes. However, this idea contrasts sharply with the traditional model of a default router.

One resolution of this problem might be to consider the entire ad hoc domain as a "single hop" from the point of view of Internet connectivity [23]. This view may be considered analogous to the way BGP characterizes an entire administrative system (AS) as a single hop in its route advertisements. However, given that multihop wireless connectivity incurs significant overhead, it is more important to minimize unnecessary hops in an ad hoc network than in a high-speed wired AS. Counting each hop and generalizing the meaning of a default router has the effect of defining the previously noted problems out of the existence. Moreover, it simplifies the choice between several possible points of attachment to the Internet (i.e., several default routers).

If we can agree that we can find a default router for an ad hoc node within an ad hoc network, then we may as well consider how to make the default router a *foreign agent* for Mobile IP. Then every node in the ad hoc cloud can appear to be as accessible as if it were still located on its home network. This has been investigated by multiple researchers [30,31]. In the Lei and Perkins project, they found that the route management for ad hoc networks had to be carefully distinguished from that for default routers (i.e., the foreign agent in Mobile IP). This same problem surfaces for any attempt to maintain routes to one of several interchangeable service points even if the service is something other than the offer of a default route.

Security: Clearly, security has so far not been satisfactorily investigated for ad hoc network protocols. Security for any routing protocol is always difficult, mainly because of the difficulties of key distribution and refresh. The topic of security for ad hoc protocols is impressive by its almost total absence. It is not clear whether public key methods or symmetric key methods are more unwieldy in ad hoc networks because both seem so difficult. Use of public key infrastructure (PKI) algorithms have disadvantages stemming from using algorithms that are more CPU intensive than the symmetric key algorithms. Furthermore, it may be difficult to make good use of certificate revocation lists (CRLs) in an ad hoc network without an infrastructure. Symmetric key algorithms, however, rely on secure distribution of secret keys shared between partners, which amounts to a key distribution problem that scales as the square of the number of partners.

Multicast data distribution in ad hoc networks is another important consideration with security implications. There will always be many different trust relationships between neighbors in any large community and many different reasons for personal relationships. As groups of people collaborate and wish to establish private video distribution channels, multicast will become crucial. Given any reasonable solution to the QoS control problem, an adaptation of that solution to multicast will probably be realized without too much procrastination. However, thinking about keys for multicast groups is almost beyond the pale given the current state of the art.

Power Control: There are two aspects to power control for wireless mobile nodes: Reducing power to the communications interfaces and entering *sleep mode* are both valuable ways to extend battery life for mobile units. These power-saving techniques are crucial to the viability of existing cellular telephones and other wireless PDAs. However, both of these techniques have the effect of making communication with the mobile device more difficult.

To enable sleep mode with wireless devices, the ad hoc routing algorithm will require some scheduling so that the wireless nodes can determine when to exit sleep mode and start listening for messages. Alternatively, a special signaling channel could wake up the wireless nodes.

The scheduling strategy requires synchronization between the wireless nodes, but achieving and maintaining synchronization is a problem that deserves significant study. For example, what happens when two ad hoc clouds collide? Changing the routing paths seems fairly straightforward, especially with an on-demand protocol. However, changing the synchronization is a whole different story.

The alternative strategy of reserving a special signaling channel is used in modern cellular telephone networks in conjunction with a paging subsystem. Besides requiring that sleeping nodes wake up, often unnecessarily, to process signals that are actually targeted to some other node, this method requires that some bandwidth be reserved just for paging, which reduces the amount of bandwidth available for data transfers. To further complicate matters, hybrid approaches that combine the advantages and disadvantages of both paging and synchronized sleep modes are possible.

In some wireless systems (notably CDMA), it is possible to vary the transmission power between stations so that the signal is just powerful enough to be correctly decoded at the receiver. Reducing the transmission power has at least these advantages:

- Power saving at the mobile nodes
- Noise reduction at all other neighboring station
- Increased frequency re-use for the transmission channel

These are powerful incentives for introducing power control into the system design for wireless network nodes.

There is an interaction between transmission power control and routing - increasing the transmission power at a network node is likely to increase the number of other nodes that are directly reachable. Thus, it is possible that some routing paths will be available only at relatively high transmission power levels; sometimes-high power is required before any routing path becomes available. One can imagine algorithms that attempt to stabilize the transmission power at a threshold level that just barely enables data transfer, but then these minimally powered links might be much more easily broken or vulnerable to errors and other disruptions.

Abstractly, power control seems to belong to layer 2, but it also has direct effects on the network layer protocol (i.e., IP). This is a puzzle that still needs to be stored out. From this perspective, transmission power control is slightly more complex than handling sleep modes, but given the complication introduced by synchronization, it seems that both are quite challenging.

AODV and DSR Improvement: Because AODV and DRS uses a broadcast route discovery mechanism, it may not be scalable to networks with large number of nodes (e.g., tens of thousands of nodes). The scalability of the algorithm will likely be improved if it is run over a clustering algorithm. As nodes across the network are grouped in to clusters and clusterheads, a hierarchy is formed whereby broadcasts are not flooded across every node in the network, but instead across every clusterhead in the network, thereby reducing the bandwidth consumed by such broadcasts. This hierarchy can easily be extended into a multilevel hierarchy that grows dynamically with the number of nodes in the network.

ABR Improvement: In ABR, while high associativity enhances communication capability, an increase in the number of active MHs in a wireless cell can cause greater contention for the available wireless bandwidth, resulting in lower throughput per MH. A suggestion for an environment, which is congested

with MHs, is to dynamically adjust the transmission power of each MH such that both the cell size and the number of neighbors are reduced in order to achieve a reasonably high throughput while still maintaining acceptable routing performance.

Addressing QoS: Although the ABR protocol is not designed to support QoS guarantees, it does distinguish the qualities among possible routes. The ABR route selection algorithm presented earlier considers routes with the highest degree of association stability and acceptable route relaying load as the more important QoS metrics, followed by minimum hop routes and routes with minimum cumulative forwarding delay. However, the order of 'route-filtering' in ABR route selection can be changed in accordance to application QoS requirements. If minimum cumulative forwarding delay and throughput are regarded as more important factors, than these metrics will overwrite the others.

The specification of the order of QoS importance has to be mapped to the underline routing protocol in some manner. After the mapping, one possible solution is to append such QoS requirements into the BQ and LQ control packets during the full and partial route discovery processes so that the destination can be informed of the desired quality of service requirements. In particular, during a RRC process, it is desirable for the pivoting node to retain the user specified QoS requirements that it has learned during the processing of the BQ packet so that this information can be appended in the LQ packets to be broadcast.

Summarizing, the ABR routes selection algorithm is dynamic and the ABR protocol can be improved to take into consideration a users QoS requirements.

ODMRP Improvement: As suggested in [26], the scalability of ODMRP can be improved by implementing the dominant pruning approach for improving the flooding of packets. In dominant pruning, the range of neighborhood information is extended to two-hop neighbors versus only directly connected nodes in self-pruning. The sender selects adjacent nodes that should relay the packet to complete the broadcast. The IDs of selected adjacent nodes are recorded in the packet as a forward list. This process is repeated until the broadcast is completed. By using techniques such as this, one could decrease the congestion and control overhead caused by flooding packets.

MAODV Improvement: The primary area of improvement is the fragility of the bi-directional shared tree causing poor packet delivery ratios. A possible improvement can be to build redundant or backup links that can be used in the event of a primary link failure. Currently, when a node receives a RREP after it has already activated a route, it will discard the message. Instead of discarding it, this can be used as a redundant link in the event of a link failure. This will save having to broadcast a Join RREQ in order to repair a broken link. That can be left as a last resort when there are no redundant links available.

QoS Improvement: Currently, there is trend towards an adaptive QoS approach over the traditional resource reservation. Very recently [6], Chakrabarti et. al published some QoS issues. The techniques they applied for QoS-preserving QoS routing detect broken routes and then either repair the broken route or reroute the flow on an alternate route with the desired QoS. If a node detects a link failure, it may attempt to repair the route by finding another node in such a way that the new path satisfies the QoS requirement between the source and the destination. If no such route segment can be found, the node (who detected route failure) notificies the source that the route is broken. Depending on the network policy, a node may send the notification of route unavailability to the source without attempting to repair the route. When the source receives the notification of route unavailability, it seeks an alternate route with the same characteristics. If found, the flow is rerouted to it. Failure to find an alternate route, the QoS route is declared unavailable and the associated resources released. I think, it could be good idea to introduce some sort of mechanism to each node that can create and engineer the path. The basic idea is that, if a node detects a route failure, it will attempt to repair it. If it fails to repair, instead of notifying the source, the nodes try to discover the alternative route from that point. In both cases, QoS requirements must be satisfied between source and destination. Time-stamps might also be necessary at each node so that a packet can get delivered prior to receiving another packet from the sender to avoid congestion. If nodes can have these capabilities, it can prioritize the packets as well as minimize the delay.

10 Conclusion

This report provides a description of several existing unicast and multicast routing protocols for ad hoc wireless networks and classified them according to their basic scheme (i.e. reactive, proactive, hybrid or flooding). I briefly, highlighted their novel concepts, strengths, weaknesses, characteristics, performance and comparison. Finally, I have identified some open issues and suggestions for improvements. To date, there is no best ad hoc routing protocol suitable for all circumstances, each protocol has definite advantages and disadvantages, and is well suited for certain situations. It is currently impossible to quantitatively compare and contrast most ad hoc routing protocols. For the most part, simulations of protocols have been done independent of one another and results cannot be compared. In order to begin filtering out inferior protocols and highlight the superior concepts, significant simulation work needs to be done in a coordinated, consistent fashion. Of the protocols described in this report, AODV and DSR have overall exhibited a good performance also when mobility is high. DSR is however based on source routing, which means that the byte overhead in each packet can affect the total byte overhead in the network quite drastically when the offered load to the network and the size of the network increases. In these situations, a hop-by-hop based routing protocol like AODV is more desirable. One advantage with the source routing approach is however that in its route discovery operation it learns more routes. Source routing is however not desirable in ordinary forwarding of data packets because of the large byte overhead. Apart from the suggestions in chapter 9, a combination of AODV and DSR could therefore be a solution with even better performance than AODV and DSR. The field of ad hoc mobile networks is rapidly growing and changing, and while there are still many challenges that need to be met, it is likely that such networks will see widespread use within the next few years.

References

[1] A. Ephremides, J.E. Weiselthier, and D. J. Baker, "A Design Concept for Reliable Mobile Radio Networks with Frequency Hopping Signaling." Proceedings of the IEEE , vol. 75, no. 1, pp.56-73, January 1987.

[2] A. K. Parekh, "Selecting Routers in Ad Hoc Wireless Networks." In Proceedings of the SBT/IEEE International Telecommunications Symposium, August 1994.

[3] A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen, "Scalable Routing Strategies for Ad Hoc Wireless Networks" IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks, Vol. 17, No. 8, Aug. 1999, pp.1369-79.

[4] C. E. Perkins, "Mobile IP. IEEE Communication Magazine" vol. 3, no.5, pp. 84-99, May 1997.

[5] C.E. Perkins, P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers", Proceedings of the SIGCOMM '94 Conference on Communications, Architectures, Protocols and Applications, pp. 234-244, August, 1994.

[6] C.E. Perkins, E.M. Royer, and S.R. Das, "Ad Hoc On Demand Distance Vector (AODV) Routing". Internet Draft, draft-ietf-manet-aodv-07.txt, November 24, 2000. Work in progress.

[7] C.E. Perkins, E.M. Royer, Samir R. Das, and M.K. Marina. "Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks." IEEE Personal Communications, Volume 39 Issue, February 2001, pp. 16-28.

[8] C.E. Perkins, and E.M. Royer, "Ad Hoc On Demand Distance Vector (AODV) Routing," In Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, pp. 90-100, New Orleans, LA, February 1999.

[9] C.E. Perkins, and E.M. Royer, "Ad Hoc On Demand Distance Vector (AODV) Routing,"

[10] C.-K Toh, "A Novel Distributed Routing Protocol To Support Ad Hoc Mobile Computing" Proc. 1996 IEEE 15th Annual International Phoenix Conference Computers and Communication, Mar. 1996, pp. 480-486.

[11] C-K. Toh, "Associativity-Based Routing for Ad-Hoc Mobile Networks," Wireless Personal Communications, Vol. 4, No. 2, March 1997, pp. 103-139.

[12] C.-T Toh, " Wireless ATM and Ad-Hoc Networks", Protocols and Architecture, Kluwer Academic Publishers, 1997.

[13] D. Bertsekas and R. Gallager, "Data Networks." Princeton-Hall, Englewood Cliffs, N.J., 1987, pp.297-333.

[14] D.J. Baker and A. Ephremides, "A Distributed Algorithm for Organizing Mobile Radio Telecommunication Networks." In Proceedings of the Second International Conference on Distributed Computer Systems, April 1981, 476-483.

[15] D.J. Baker and A. Ephremides, "The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm." IEEE Transactions on Communications COM-vol.29 no.11, pp.1694-1701, November 1981.

[16] D. B. Johnson, "Routing in Ad Hoc Networks of Mobile Hosts." Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, December, 1994.

[17] D. B. Johnson, D.A. Maltz, "Dynamic Source Routing in Ad Hoc Wirless Networks", Mobile Computing, T. Imielinski and H. Korth, Eds., Kulwer, 1996, pp. 152-81.

[18] D.A. Maltz, J. Broch, and D.B. Johnson, "Lessons from a Full-Scale Multihop Wireless Ad Hoc Network Testbed", IEEE Personal Communications, Volume 39 Issue , February, 2001, pp. 8-15.

[19] E. M. Royer and C.-K. Toh. "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks." IEEE Personal Communications Magazine, Vol. 6, No. 2, pp. 46-55, April 1999.

[20] E.M. Royer, Routing in Ad Hoc Mobile Networks: On-Demand and Hierarchical Strategies". PhD Dissertation, Electrical and Computer Engineering, University of California, Santa Barbara, December 2000.

[21] E.M. Royer, and C.E. Perkins, "Multicast Ad Hoc On Demand Distance Vector (MAODV) Routing." Internet Draft, draft-ietf-manet-maodv-00.txt, July 15, 2000. Work in progress.

[22] E.M. Royer, and C.E. Perkins, "Multicast Operation of the Ad Hoc On Demand Distance Vector (MAODV) Routing.Protocol." In proceedings of the 5th ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM), pp. 207-218, Seattle, WA, August 1999.

[23] E.M. Royer, and C.E. Perkins, "Transmission Range Effects on AODV Multicast Communication." ACM/Baltzer (accepted for publication), 2001.

[24] E.L. Madruga and J.J. Garcia-Luna_Aceves, "Multicasting Along Meshes in Ad Hoc Networks," In Proceedings of IEEE ICC'99, Vancouver, Canada, June 1999, pp.314-318.

[25] E. Bommaiah, A. McAuley, R. Talpade, and M. Liu, "AMRoute: Ad Hoc Multicast Routing Protocol." Internet Draft, draft-talpade-manet-amroute-00.txt, August 6, 1998.

[26] E.Cheng, "On-Demand Multicast Routing in Mobile Ad Hoc Networks." M. Engineering thesis, Department of Systems and Computer Engineering, Carleton University, Ottawa, January 2001.

[27] G. Pei, M. Gerla, and T.-W. Chen, "Fisheye State Routing in Mobile Ad Hoc Networks". Internet Draft, draft-ietf-manet-fsr-00.txt, November, 2000.

[28] G. Pei, M. Gerla, X. Hong and C-C Chiang, "A Wireless Hierarchical Routing Protocol with Group Mobility", Proceedings of IEEE SICON '98, University of California, LA.

[29] G. Pei, M. Gerla, and T.-W. Chen, "Fisheye State Routing in Mobile Ad Hoc Networks". In proceedings of the IEEE International Conference on Communications (ICC), pp.70-74, New Orleans, LA, June, 2000.

[30] H. Lei and C.E. Perkins, "Ad Hoc Networking with Mobile IP." In Proceedings of the Second European Personal Mobile Communications Conference, October 1997, pp.197-202.

[31] J. Broach, D.A. Maltz, and D.B. Johnson. "Supporting Hierarchy and Heterogeneous Interfaces in Multi-Hop Wireless Ad Hoc Networks". In Proceedings Of the Workshop on Mobile Computing, IEEE, Perth, Western Australia, June 1999.

[32] J. Broch, D.A. Maltz, D.B. Johnson, Y.-C. Hu, J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols", Proceedings of the Fourth Annual ACM/IEEE IC Mobile Computing and Networking (MobiCom '98), October 25, 1998.

[33] J. Broach, D.B. Johnson , and D.A. Maltz, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks." Internet Draft, draft-talpade-manet-dsr-03.txt, October 1999. Work in Progress.

[34] J.J. Garcia-Luna_Aceves and E.L. Madruga, "The Core-Assisted Mesh Protocol", IEEE Journal on Selected Areas in Communications, vol. 17, no. 8, August 1999, pp. 1380-1394.

[35] J. Jubin and J.D. Tornow, "The DARPA Packet Radio Network Protocols." In proceedings of the IEEE, vol. 75, no. 1, January 1987, pp.21-32.

[36] J. M. Jaffe and F. M. Moss, "A responsive distributed routing algorithm for computer networks." IEEE Transactions on Communications, 30(7): pp.1758-1762, July 1982.

[37] J.M. McQuillan, I. Richer, and E.C. Rosen. The New Routing Algorithm for the ARPANET. IEEE Transactions on Communications COM-28(5): 711-719, May 1980.

[38] J.J. Garcia-Luna_Aceves and E.L. Madruga, "A Multicast Routing Protocol for Ad Hoc Networks", In Proceedings of IEEE INFOCOM'99. New York, NY, March 1999, pp. 784-792.

[39] Larry L. Peterson and Bruce S. Davie, "Computer Networks - A System Approach". San Francisco, Morgan Kaufmann Publishers Inc. ISBN 1-55860-368-9.

[40] L. Kleinrock and K. Stevens, "Fishey: A Lenslike Computer Display Transformation." Technical report, UCLA, Computer Science Department, 1971.

[41] M. DeHoust, "Routing in Mobile Ad Hoc Networks", Individual Project Report. April 30, 2000.

[42] M. Gerla and J.T.-C. Tsai. Multicluster, Mobile Multimedia Radio Network. Wirless Networks 1(3): 255-265, October 1995.

[43] M. Ohta and J. Crowcroft, Static Multicast, " Internet Draft, draft-ohta-static-multicast-01.txt, October 1998.

[44] M. S. Corson and A. Ephremides, "A Distributed Routing Algorithm for Mobile Wireless Networks." ACM/Baltzer Wireless Networks J., vol. 1, no. 1, February 1995, pp.61-81.

[45] M.S. Corson and J. Macker. "Mobile Ad-hoc Networking (MANET) Routing Protocol Performance Issues and Evaluation Consideration." IETF RFC 2501, Jan 1999.

[46] P. Johansson, T. Larsson, N. Hedman, and B. Mielczarek, "Routing Protocols for Mobile Ad Hoc Networks - a Comparative Performance Analysis." In proceedings of the 5th International Conference on Mobile Computing and Networking (ACM MOBICOM'99), pp. 195-206, August 1999.

[47] P.M. Merlin and A. Segall, "A False Distributed Routing Protocol." IEEE Transactions on Communications COM-27, pp.1280-1287, September 1979.

[48] R. Dube, C. Rais, K.-Y. Wang, S. Tripathi, "Signal Stability based Adaptive Routing (SSA) for Ad-Hoc Mobile Networks." IEEE Personal Communications, February, 1997.

[49] R. Määttä, "Wireless Ad Hoc Routing Protocols, a Taxonomy". Defence Forces Research Institute of Technology, Electronics and Information Technology Section, May 11, 2000.

[50] S. Murthy and J.J. Garcia-Luna-Aceves, A Routing Protocol for Packet Radio Networks", Proceeding, First International Conference on Mobile Vomputing and Networking (ACM Mobicom), Barkley, California, November 13-15, 1995.

[51] S. Chakrabarti, A. Mishra, "QoS Issues in Ad Hoc Wirless Networks", IEEE Communications Magazine, pp. 142-150, February 2001.

[52] S. J. Lee, M. Gerla, and C.-K Toh, "A simulation Study of Table-Driven and On-Demand Routing Protocols for Mobile Ad Hoc Networks," IEEE Network, vol. 13, no. 4, July 1999, pp.48-54.

[53] S.-J. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia, "A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols." In Proceedings of the IEEE Conference on Computer Communications (INFOCOM); pp. 565-574, Tel Aviv, Isreal, March 2000.

[54] S.-J. Lee, W. Su, and M. Gerla, " On Demand Multicast Routing Protocol (ODMRP) for Ad Hoc Networks." Internet Draft, draft-ietf-manet-odmrp-02.txt, January, 2000. Work in progress.

[55] S. H. Bae, S.-J Lee, W. Su, and M.Gerla, "The Design, Implementation, and Performance Evaluation of the On-Demand Multicast Routing Protocols in Multihop Wirless Networks," IEEE Network, vol. 13, no. 4, January/February 2000, pp.70-77.

[56] S. H. Bae, S.-J Lee, and M.Gerla, " Multicast Routing Implementation and Validation in an Ad Hoc Network Testbed." Research report funded by Intel and DARPA under contract DAAB07-97-C-D321.

[57] S. Murthy and J.J. Garcia-Luna-Aceves, "An Efficient Routing Protocol for Wireless Networks", ACM Mobile Networks and App. J., Special Issue on Routing in Mobile Communication Networks, Oct. 1996, pp. 183-97.

[58] S. R. Das, R. Castaneda, and J. Yan, "Simulation Based Performance Evaluation of Mobile, Ad hoc Network Routing Protocols", Proceeding of the 7th International Conference on CCN, Oct. 1998.

[59] T.-W. Chen and Mario Gerla, "Global State Routing: A New Routing Scheme for Ad Hoc Wirless Networks" Proc. IEEE ICC'98, 5 pages.

[60] V. D. Park and M. S. Corson, "A Performance Comparison of TORA and Ideal Link State Routing", Proceedings of IEEE Symposium on Computers and Communication '98, June, 1998.

[61] V. D. Park and M. S. Corson, "A Highly Adaptive Distributed Routing Algorithm For Mobile Wireless Networks." Proceedings of IEEE Conference on Computer Communications (INFOCOM), pp.1405-1413, Kobe, Japan, April 1997.

[62] V. D. Park and M. S. Corson, "Temporary-Ordered Routing Algorithm (TORA) Version 1, Functional Specification." Internet Draft, draft-ietf-manet-tora-03.txt, November 24, 2000. Work in Progress.

[63] W. Bruce Puckett, "An Energy-Efficiency and Performance Comparison of ABR and DSR", Individual Project report, ECPE 6504: Wireless Networks and Mobile Computing.

[64] Z. Haas, and M. Pearlman, "Determining the Optimal Configuration for the Zone Routing Protocol." IEEE Journal on Selected Areas of Communications (Special issue, Ad Hoc Networks), 17 (8), August 1998.

[65] Z. J. Haas and M.R. Pearlman, "The Performance of Query Control Schemes for the Zone Routing Protocol," ACM SIGCOMM'98, September 1998, pp.167-177

[66] Z. J. Haas and M.R. Pearlman, " The Zone Routing Protocol (ZRP) for Ad Hoc Networks," Internet Draft, draft-ietf-manet-zone-02.txt, June, 1999. Work in Progress.

Appendix A – Abbreviations and Terminology

A1. Abbreviations

ABR	Associativity-Based Routing
AMRoute	Ad hoc Multicast Routing
AP	Access Points
AODV	Ad-hoc On-demand Distance Vector
BQ	Broadcast Query
BRP	Bordercast Resolution Protocol
BS	Base Station
CAMP	Core Assisted Mesh Protocol
CBT	Core Based Tree
CBR	Constant Bit Rate
CC	Computation Complexity
CEJ	Core Explicit Joins
CGSR	Clusterhead Gateway Switch Routing
CLR	Clear
CPU	Central Processing Unit
DSDV	Destination Sequenced Distance Vector
DBF	Distributed Bellman Ford algorithm
DNS	Domain Name Service
DRP	Dynamic Routing Protocol
DSR	Dynamic Source Routing
DVMRP	Distance Vector Multicast Routing Protocol
ERS	Expanded Ring Search
ET	Early Termination
FIFO	First In First Out
FSR	Fisheye State Routing
GPS	Global Positioning System
GRPH	Group Hello
GSR	Global State Routing
HSR	Hierarchical State Routing
HID	Hierarchical ID
IARP	Intrazone Routing Protocol
ICMP	Internet Control Message Protocol
IERP	Interzone Routing Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force

ILS	Idealized Link-State protocol
IMEP	Internet MANET Encapsulation Protocol
IN	Intermediate Node
IP	Internet Protocol
LAN	Local Area Network
LAM	Lightweight Multicast Protocol
LCC	Least Cluster Change
LQ	Local Query
LRU	Least Recently Used
LS	Link State
MAC	Media Access Protocol
MACT	Multicast Route Activation
MANET	Mobile Ad-hoc NETworks
MAODV	Multicast Ad hoc On-Demand Vector routing protocol
MRL	Message retransmission list
NSMP	Neighbour Supporting Multicast Protocol
ODMRP	On-Demand Multicast Routing Protocol
PDA	Personal Digital Assistant
ZRP	Zone Routing Protocol
PIM-DM	Protocol Independent Multicast Dense Mode
PIM-SM	Protocol Independent Multicast Sparse Mode
QoS	Quality of Service
RPF	Reverse Path Forwarding
RREQ	Route Request
RREP	Route Reply
RRC	Route Reconstruction
RT	Routing Table
SRP	Static Routing Protocol
SSR	Signal Stability-Based Adaptive Routing protocol
SST	Signal Stability Table
TCP	Transmission Control Protocol
TORA	Temporally Ordered Routing Algorithm
TT	Topology Table
TTL	Time To Live
WAM	Wireless Adaptive Mobility
WRP	Wireless Routing Protocol

A.2 Terminology Used

Asymmetric: A link with transmission characteristics those are different of the transmitter and receiver. For instance, the range of one transmitter may be much higher than the range of another transmitter on the same medium. The transmission between the two hosts will therefore not work equally well in both directions.

Bandwidth: Total link capacity of a link to carry information (typically bits).

Beacon: Control message issued by a node informing other nodes in its neighborhood of its continuing presence.

Channel: The physical medium is divided into logical channel, allowing possibly shared uses of the medium. Channels may be made available by subdividing the medium into distinct time slots, distinct spectral bands, or decorrelated coding sequences.

Cluster: A group of nodes typically in range of each other, where one of the nodes is elected as the cluster head. The cluster head ID identifies the cluster. Each node in the network knows its corresponding cluster head(s) and therefore knows which cluster(s) it belongs to.

Convergence: The process of approaching a state of equilibrium in which all nodes in the network agree on a consistent state about the topology of the network.

Flooding: The process of delivering data or control messages to every node within the any data network

Host: Any node that is not a router.

Interface: A nodes attachment to a link.

Link: A communication facility or medium over which nodes can communicate at the link layer.

Loop free: A path taken by a packet never transits the same intermediate node twice before arrival at the destination.

MAC-layer address: An address (sometimes called the link address) associated with the link interface of a node on a physical link.

Next hop: A neighbor, which has been designated to forward packets along the way to a particular destination.

Neighbor: A node that is within transmitter range from another node on the same channel.

Node: A device that implements IP.

Node ID: Unique identifier that identifies a particular node.

Proactive: Tries to maintain the routing map for the whole network all the times.
Reactive: Calculates route only upon receiving a specific request.
Router: A node that forwards IP packets not explicitly addressed to itself. In case of ad hoc networks, all nodes are at least unicast routers.

Routing table: The table where the routing protocols keep routing information for various destinations. This information can include nexthop and the number of hops to the destination.

Scalability: A protocol is scalable if it is applicable to large as well as small populations.

Source route: A route from the source to the destination made available by the source.

Symmetric: Transmission between two hosts works equally well in both directions.

Throughput: The amount of data from a source to a destination processed by the protocol for which throughput is to be measured for instance, IP, TCP, or the MAC protocol.