ROUTING WITH LOAD BALANCING IN WIRELESS AD HOC NETWORKS

BY

ZHUNYAN (AUDREY) ZHOU

A thesis submitted to the

Department of Computing and Information Science

in conformity with the requirements for

the degree of Master of Science

Queen's University

Kingston, Ontario, Canada

January 2001

Abstract

An ad hoc wireless mobile network is an infrastructureless mobile network that has no fixed routers; instead, all nodes are capable of movement and can be connected dynamically in an arbitrary manner. In order to facilitate communication of mobile nodes that may not be within the wireless range of each other, an efficient routing protocol is used to discover routes between nodes so that messages may be delivered in a timely manner.

In this thesis, we present a Load-Balanced Ad-hoc Routing (LBAR) protocol for communication in wireless ad-hoc networks. LBAR defines a new metric for routing known as the degree of nodal activity. In LBAR, a route is selected based on activity of nodes. The routes selected are, therefore, likely to have least traffic load and transmit traffic in a timely manner. LBAR has two phases, namely, *route discovery* and *path maintenance*. The route discovery phase broadcasts setup messages to the destination,

which is responsible for collecting routing information and selecting best routes. Whenever a link failure is detected, path maintenance is initiated to patch up by detouring traffic to the destination. A comprehensive simulation model was conducted to study the performance of the proposed scheme. Performance results show that LBAR outperforms existing ad-hoc routing protocols in terms of packet delivery and average end-to-end delay.

Key words: Wireless Ad-hoc Network, Routing, Load Balancing, Performance Evaluation, Path Discovery, and Path Maintenance

Acknowledgements

I am particularly grateful to my advisor, Dr. Hossam Hassanein. His support, patience, and guidance made this work possible. I would also like to thank all members of the examination committee for their suggestions and comments and their time reviewing this dissertation.

I am also indebted to my parents for their endless love, support, encouragement and understanding that provided me with the motivation and endeavor to accomplish this piece of work.

The financial support provided by the department of Computing and Information Science at Queen's University and Communication and Information Technology Ontario (CITO) is greatly appreciated.

Last but not least, I would like to express my special thanks to my dearest friends. They made my first stay in Canada such an enjoyable and unforgettable experience. They gave me best-effort support and warm-hearted care. They are one of my greatest pride I have achieved in Canada.

List of Acronyms

ABR	Associativity Based Routing
AODV	Ad-hoc On-Demand Distance Vector
CAC	Call Admission Control
CBR	Continuous Bit Rate
CGSR	Cluster-head Gateway Switch Routing
CSMA/CD	Carrier Sense Multiple Access / Collision Avoidance
DBF	Distributed Bellman-Ford
DSDV	Destination-Sequenced Distance Vector
DSR	Dynamic Source Routing
DV	Distance Vector
FSR	Fisheye State Routing
GSR	Global State Routing
IETF	Internet Engineering Task Force

LBAR	Load-Balanced Ad-hoc Routing
LSR	Link State Routing
MAC	Medium Access Control
MANET	Mobile Ad-hoc NETworking working group
RREP	Route REPly
RREQ	Route REQuest
QoS	Quality of Service
VBS	Virtual Base Stations
WRP	Wireless Routing Protocol
ZRP	Zone Routing Protocol

Contents

List of Figures

Chapter 1

Introduction

Since their emergence in the 1970's, wireless networks have become increasingly popular in the computing industry. This is particularly true within the past decade, which has seen wireless networks being adapted to enable mobility. There are currently two variations of mobile wireless networks. The first is known as the infrastructure-based networks, which require a preconfigured, fixed infrastructure, e.g., cellular networks [10]. In infrastructure-based networks, the whole service area is divided into several smaller service regions called cells. In each cell, at least one base station is allocated to provide network service to mobile nodes in the cell. A mobile node within these networks connects to, and communicates with the nearest base station that is within its communication radius. The connections among base stations are usually provided by a high speed wired backbone. Because of the wired backbone, wireless communications in

infrastructure-based network only exist between a mobile node and the associated base station.

The second type of mobile wireless network is the infrastructureless mobile network, commonly known as an ad hoc network [11]. Infrastructureless networks have no fixed infrastructure; all nodes are capable of movement and can be connected dynamically in an arbitrary manner and form a temporary network without the aid of any established infrastructure or centralized administration. Nodes of these networks function as routers, which can discover and maintain routes to other nodes in the network. The interest in ad hoc network is fuelled by its unique characteristics of independence of fixed infrastructure and adaptation in dynamic environment. Thus, such networks can provide a more flexible service, for example, in a rural area or battlefield where wireless access to a wired backbone is either ineffective or impossible.

A critical challenge in the design of ad hoc networks is the development of efficient routing protocols that can provide high-quality communication between two mobile nodes. If only two nodes, located closely together, are involved in the ad hoc network, no real routing protocol or routing decisions are necessary. In many ad hoc networks, however, two nodes that want to communicate may not be within the wireless transmission range of each other. Hence, a routing decision must be made to route packets from source to destination through other nodes participating in the ad hoc

network. For example, in the network illustrated in Figure 1.1, mobile node C is not within the transmission range of node A (indicated by the circle around A) and node A is not within the transmission range of node C. If A and C wish to exchange packets, they may in this case enlist the services of node B to forward packets for them, since B is within the overlap between A's range and C's range.

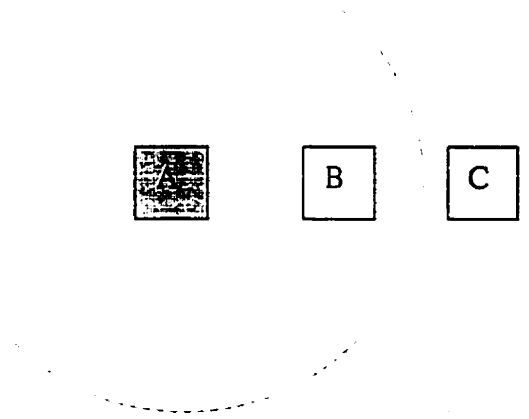

Figure 1.1 A simple ad hoc network of three wireless mobile nodes

Numerous routing protocols have been developed for ad hoc mobile networks. These protocols may generally be categorized as table-driven and on-demand routing. Table driven routing protocols [1][2][3][4][15] attempt to maintain consistent, up-to-date routing information in each node by propagating updates throughout the network. Although a route to every other node is always available, such protocols incur substantial

signaling traffic and power consumption. Since both bandwidth and battery power are scarce resources in mobile computers, this becomes a serious limitation to table-driven routing protocols. On the other hand, on-demand routing protocols [5][6][7][8][9] overcome this limitation. This type of routing protocols do not maintain all up-to-date routes at every node, whereas create routes only when desired by the source node. When a source has a packet to transmit, it invokes a route discovery mechanism to find the path to the destination. The route remains valid until the destination is reachable or until the route is no longer needed. AODV [9] and DSR [7][8] are the two most prominent ad hoc on-demand routing protocols, nominated as strong candidates for standardization by IETF (Internet Engineering Task Force). In fact, on-demand routing is dominating the tendency for wireless ad hoc communication.

However, none of the proposed on-demand ad hoc routing protocols consider the issue of load balancing to attempt to evenly distribute data traffic in the network and thus lower the average end-to-end delay. Due to lack of load balancing mechanism, data traffic may overwhelm some nodes and bring about congestion while some other nodes remain idle. As a result, the overwhelmed nodes would ultimately become a bottleneck and delay the transmission of traffic. Because of this, the performance of AODV and DSR is degraded by the high end-to-end delay. Hence, in this thesis, we propose an efficient routing protocol for wireless ad hoc networks, namely, the Load-Balanced Ad-hoc Routing (LBAR) protocol. The proposed scheme is intended to route data packets circumventing

congested paths so as to balance traffic load over the network and lower end-to-end delay. Additionally, the protocol demonstrates quick response to link failures incurred by topology changes in the ad-hoc network and thereby improves data delivery reliability. As will be shown in the simulation results (Chapter 4), LBAR outperforms both AODV and DSR in terms of packet delivery fraction and average end-to-end delay.

This thesis is organized as the following. The next chapter surveys existing routing algorithms for wireless networks. In Chapter 3, the details of the proposed LBAR scheme are described. Simulation results and analysis are reported in Chapter 4. Finally, Chapter 5 presents conclusions and future work.

Chapter 2

Previous Work

Routing is the function in the network layer which determines the path from a source to a destination for the traffic flow. In wireless ad-hoc networks, due to the node mobility, network topology may change from time to time. So it is critical for the routing protocol to deliver data packets efficiently between source and destination. This chapter presents a literature review of routing in today's wireless ad-hoc networks. These routing protocols can be divided into two categories: table-driven and on-demand routing based on when and how the routes are discovered. In table-driven routing protocols consistent and up-to-date routing information to all nodes is maintained at each node whereas in on-demand routing the routes are created only when desired by the source node. Section 2.1 discusses current table-driven protocols, while Section 2.2 describes those protocols which are classified as on-demand. Hybrid routing protocols are discussed in Section 2.3. Finally, a summary is given in Section 2.4.

2.1 Table Driven Routing Protocols

Table-driven routing protocols attempt to maintain consistent, up-to-date routing information from each node to every other node in the network. These protocols require each node to maintain one or more tables to store routing information, and they respond to changes in network topology by propagating updates throughout the network in order to maintain a consistent network view. These routing protocols differ in the method by which topology change information is distributed across the network and the number of necessary routing-related tables.

2.1.1 Distributed Bellman-Ford (DBF)

Many existing routing schemes for ad hoc wireless network are based on the distributed Bellman-Ford's (DBF) algorithm. These schemes are also referred to as distance vector (DV) schemes. In the distributed Bellman-Ford algorithm, every node i maintains a routing table which is a matrix containing distance and successor information for every destination, where distance is the length of the shortest distance from i to j and successor is a node that is next to i on the shortest path to j. To keep the shortest path information up to date, each node periodically exchanges its routing table with its neighbors. Based on the routing tables received with respect to its neighbors, node i learns the shortest distances to all destinations from its neighbors. Thus, for each destination j, node i selects a node k from its neighbors as the successor to this destination (or the next hop) such that

the distance from i through k to j will be the minimum. This newly computed information will then be stored in node i's routing table and will be exchanged in the next routing update cycle. Figure 2.1 shows an example of DV routing. Node S receives two routing tables from its neighbors node 2 and node 3. By comparison of distance field in the routing table, the path S-2-1-D is shorter than the path S-4-3-1-D. So node S selects the path S-2-1-D as the shortest path to the destination node D and identifies node 2 as its successor.

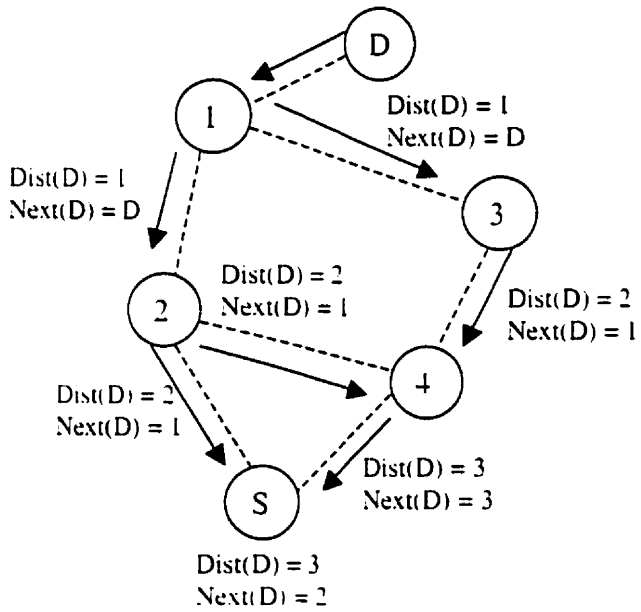

Figure 2.1 An example of Distance Vector Routing

The advantages of DBF are its simplicity and computation efficiency due to its distributed nature. However, it is well known that DBF is slow to converge when the topology changes, and that it has the tendency to create routing loops.

2.1.2 Destination-Sequenced Distance-Vector (DSDV)

The Destination-Sequenced Distance-Vector (DSDV) Routing Algorithm described in [1] is based on the idea of the classical Bellman-Ford Routing Algorithm. Every mobile node in the network maintains a routing table that lists all available destinations and the number of hops to reach the destination. Each entry is marked with a sequence number assigned by the destination node. The sequence number is used to distinguish stale routes from new ones and thus avoid the formation of loops. Routing table updates are periodically transmitted throughout the network in order to maintain table consistency. A node also transmits its routing table if a significant change has occurred in its table from the last update sent. So, the update is both time-driven and event-driven. To help alleviate the potentially large amount of network traffic that such updates can generate, the routing table updates can be sent in two ways: a *full dump* or an *incremental update*. A full dump sends the full routing table to the neighbors and could span many packets whereas in an incremental update only those entries from the routing table are sent that have a metric change since the last update. When the network is relatively stable, incremental updates

are sent to avoid extra traffic and full dumps are relatively infrequent. In a fast-changing network, incremental packets can grow big so full dumps will be more frequent.

New route broadcasts contain the address of the destination, the sequence number of the information received regarding the destination, as well as a new sequence number unique to the broadcast. The route labeled with the highest, i.e., most recent, sequence number is always used. In the event that two updates have the same sequence number, the route with the best metric, i.e., shortest route, is used in order to optimize the path.

DSDV routing is essentially a modification of the basic Bellman-Ford routing algorithm. The modifications include the guarantee of loop-free routes and a simple route update protocol. While only providing one path to any given destination, DSDV selects the shortest path based on the number of hops to the destination. DSDV provides two types of update messages, one of which is significantly smaller than the other. The smaller update message can be used for incremental updates so that the entire routing table need not be transmitted for every change in network topology. However, DSDV is inefficient because of the requirement of periodic update transmissions, regardless of the number of changes in the network topology. DSDV also requires each mobile node to maintain a complete list of routes, one for each destination within the ad hoc network. This almost always exceeds the needs of any particular mobile node.

2.1.3 Wireless Routing Protocol (WRP)

The Wireless Routing Protocol (WRP) [4] is also a table-based distance-vector routing protocol based on DBF with the goal of maintaining routing information among all nodes in the network. Each node is responsible for maintaining four tables: *distance table*, *routing table*, *link-cost table* and *message retransmission list table*. The distance table of a node x contains the distance of each destination node y via each neighbor z of x. It also contains the downstream neighbor of z through which this path is realized. The routing table of node x contains the distance of each destination node y from node x, the predecessor and the successor of node x on this path. It also contains a tag to identify if the entry is a simple path, a loop or invalid. The link-cost table of node i contains the cost of relaying information through each neighbor k, and the number of periodic update periods that have elapsed since node i received any error-free messages from k. The message retransmission list contains information to let a node know which of its neighbors has not acknowledged its update message in order to retransmit an update message to that neighbor.

Mobile nodes inform each other of link changes through the use of update messages. An update message is sent only between neighboring nodes and contains a list of updates, as well as a list of responses indicating which mobile nodes should acknowledge the update. Mobiles send update messages after processing updates from neighbors or detecting a change in a link to a neighbor. In the event of the loss of a link between two nodes, the

nodes send update messages to their neighbors. The neighbors then modify their distance table entries and check for new possible paths through other nodes. New paths are relayed back to the original nodes so that they can update their tables accordingly. Nodes learn of the existence of their neighbors from the receipt of acknowledgments 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. When a mobile node receives a hello message from a new node, that new node is added to the mobile's routing table, and the mobile sends the new node a copy of its routing table information.

WRP falls short in that it still requires significant periodic information exchange. The volume of routing overhead required to maintain shortest-path trees to all destinations will be substantial when nodes become highly mobile or the network becomes large. Consequently, protocol scalability and rapid adaptation in highly dynamic environments is unlikely.

2.1.4 Global State Routing (GSR)

Global State Routing (GSR) [3] was proposed by Chen and Gerla, with the objective of providing global topology knowledge without the overhead of a full link-state protocol. The objective is similar to WRP, however, unlike WRP, the underlying approach is not

based on adaptation of Distance Vector Routing. GSR is functionally similar to Link State Routing (LSR) [13] [14] in that it maintains a topology map at each node. The key is the way in which routing information is disseminated. The scheme adopts the idea of LSR but improves it by avoiding flooding of routing messages. In OSPF [20] - a link state routing protocol - link state packets are generated and flooded onto the network whenever a node detects a topology change. In GSR, link state packets are not flooded. Instead, nodes maintain a topology 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. The GSR periodic table exchange resembles the vector exchange in DSDV where the distances are updated according to the time stamp or sequence number assigned by the node originating the update. In GSR, link states are propagated, a full topology map is kept at each node, and shortest paths are computed using this map.

In this algorithm, each node maintains a *neighbor list*, a *topology table*, a *next hop table* and a *distance table*. A neighbor list of a node contains the list of its neighbors (here all nodes that can be heard by a node are assumed to be its neighbors.). For each destination in the ad hoc network, the topology table contains the link state information as reported by the destination. In addition, 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.

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

The drawbacks of GSR are the large size update messages, which consume a considerable amount of bandwidth and the latency of the link state change propagation, which depends on the update period. GSR has to maintain knowledge of the full network topology, which becomes unnecessary when topological changes are less frequent, and hence, the demand on such knowledge is less. Moreover, even when the network topology changes rapidly, but the active ongoing communications are not actually affected by such changes, it is again unnecessary to trigger any updates to reflect the topological variations. This is also not desirable in the case of large populations of mobile nodes.

2.1.5 Fisheye State Routing (FSR)

Fisheye State Routing (FSR) [15] is an improvement of GSR [3]. Although GSR computes accurate routing decisions using global network information, the large size of update messages in GSR wastes a considerable amount of network bandwidth. In FSR, each update message does not contain information about all nodes. Instead, it updates the network information for nearby nodes at a higher frequency than for remote nodes, which are outside the "fisheye" scope. So each node gets accurate information about neighbors and the detail and accuracy of information decreases as the distance from node increases.

Figure 2.2 shows the scope of fisheye for the center node. The large circles define the fisheye scope of the center node. The scope of fisheye is defined as the nodes that can be reached within a certain number of hops. In this case, three scopes are shown and they represent the scope of 1-hop, 2-hop and 3-hop. The center nodes have most accurate information about all nodes in the white circle. The longer the distance from the white circle, the less accurate information about all the nodes in the white circle the nodes have.

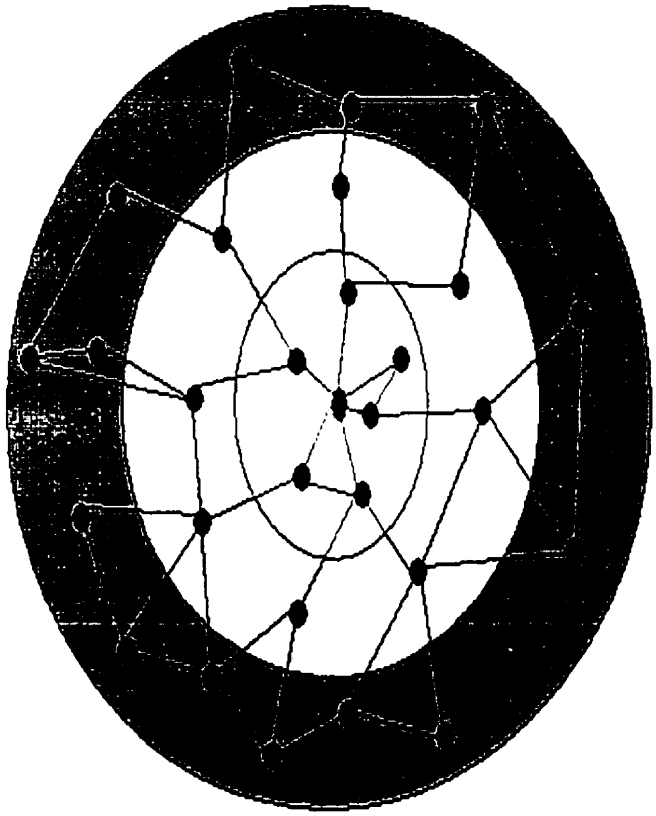

Figure 2.2 Scope of fisheye

The advantage of FSR is that it scales well to large networks by keeping link state exchange overhead low. However, as mobility increases, routes to remote destinations become less accurate, which would cause high average delay and packet loss.

2.1.6 Cluster-head Gateway Switch Routing (CGSR) protocol

The Cluster-head Gateway Switch Routing (CGSR) protocol differs from the previous protocols in the type of addressing and network organization scheme employed. Instead of a "flat" network, CGSR is a clustered multi-hop mobile wireless network with several heuristic routing schemes [2].

The mobile nodes are aggregated into clusters and a *cluster-head* is elected. All nodes that are in the communication range of the cluster-head belong to its cluster. A gateway node is a node that is in the communication range of two or more cluster-heads. 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 algorithm works in the following manner. A packet sent by a node is first routed to its cluster head, and then the packet is routed from the cluster head to a gateway and then to another cluster head, and so on until it reaches the cluster head of the destination node. The packet is then transmitted to the destination. Figure 2.3 illustrates an example of this routing scheme. Using this method, each node maintains a *cluster member table* that has mapping from each node to its respective cluster-head. Each node broadcasts its cluster member table periodically and updates its table after receiving other nodes' broadcasts

using the DSDV algorithm. In addition, each node also maintains a routing table that determines the next hop to reach the destination cluster.

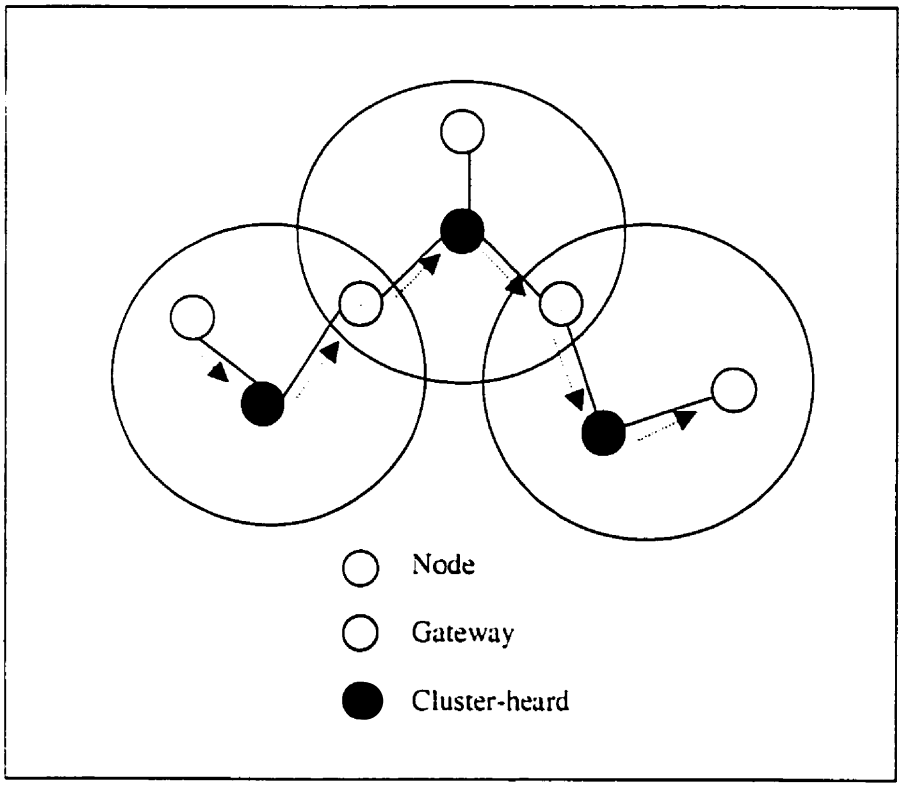

Figure 2.3 An example of Cluster-head Gateway Switch Routing (CGSR)

Upon receiving a packet, a node consults its cluster member table and routing table to determine the nearest cluster head along the route to the destination. Then it checks its routing table to find the next hop in order to reach the cluster-head and transmits the packet to that node.

A cluster head selection algorithm is utilized to elect a node as the cluster head using a distributed algorithm within the cluster. 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. In CGSR, because routing performance is dependent on the status of specific nodes (cluster head, gateway or regular), time complexity of a link failure associated with a cluster head is higher than DSDV, given the additional time needed to perform cluster head reselection.

As discussed earlier, table-driven ad hoc routing approach is similar to the connectionless approach of forwarding packets, with no regard to when and how frequently such routes are desired. It relies on an underlying routing table update mechanism that involves the constant propagation of routing information. On the other hand, because routing information is constantly propagated and maintained in table-driven routing protocols, a route to every other node in the ad hoc network is always available, regardless of whether or not it is needed. This feature, although useful for datagram traffic, incurs substantial signaling traffic and power consumption. Since both bandwidth and battery power are scarce resources in mobile computers, this becomes a serious limitation. So in recent literature, on-demand ad hoc routing has largely been advocated as a universal remedy for the ad hoc routing problem.

2.2 On-Demand Routing Protocols

A different approach from table-driven routing is on-demand routing. This type of routing is a form of dynamic resource discovery algorithm. They are similar in some respects to mobility management algorithms and similar in other respects to connection-oriented routing. Unlike table-driven routing, on-demand protocols do not rely on periodic exchange of routing information. Instead, they establish and maintain routes on a demand basis. When a node requires a route to a destination, it initiates a route discovery process within the network. 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 on longer desired. All on-demand protocols, which have been proposed for ad hoc networks specify three common operations or phases:

1. Path discovery: Search for destination and construct route.
2. Path maintenance: Detect path failure due to node mobility and attempt to re-establish path.
3. Path erasure: Delete either an entire path, or a portion of a path following node movement, when it is no longer needed, or continued use of it could result in packet looping.

On demand routing protocols differ in the mechanisms and metrics used to achieve each of these operations and the mechanism used in packet forwarding. The broad objective of all the schemes is to minimize the amount of routing protocol reaction required to adapt following node movement.

2.2.1 Dynamic Source Routing (DSR) protocol

The Dynamic Source Routing protocol (DSR) [7] [8] is an on-demand routing protocol based on the concept of source routing, which creates on-demand paths using a route query process based on broadcast of a route request packet. Once a route has been acquired by a source, the source caches that route locally until it is either informed that the route is no longer valid due to mobility along the path, or the node no longer requires the route and an inactivity timer has expired.

The protocol consists of two major phases: route discovery and route maintenance. Route discovery allows any node in the ad hoc network to dynamically discover a route to any other node in the ad hoc network, whether directly reachable within the wireless transmission range or reachable via one or more intermediate network hops through other nodes. When a mobile node has a packet to send to some destination, it first consults its route cache to determine whether it already has a route to the destination. If it has an unexpired route to the destination, it will use this route to send the packet. On the other

hand, if the node does not have such a route, it initiates route discovery by broadcasting a route *request* packet. This route request packet contains the address of the destination, along with the source node's address and a unique identification number *request id* set by the initiator from a locally maintained sequence number. Each node receiving the packet checks whether it knows of a route to the destination. If it does not, it adds its own address to the route record of the packet and then forwards the packet along its ongoing links. To limit the number of route requests propagated on the outgoing links of a node, a mobile only forwards the route request if this request has not yet been seen by the mobile and if the mobile's address does not already appear in the route record.

For example, consider the scenario in Figure 2.4, in which node 2 appends its own address to the request packet it receives from node 1. Node 5 receives two copies of the *request* packet. It appends its address to the first copy and discards the request packet from node 4 because it has previously received a request packet with the same source-request id pair.

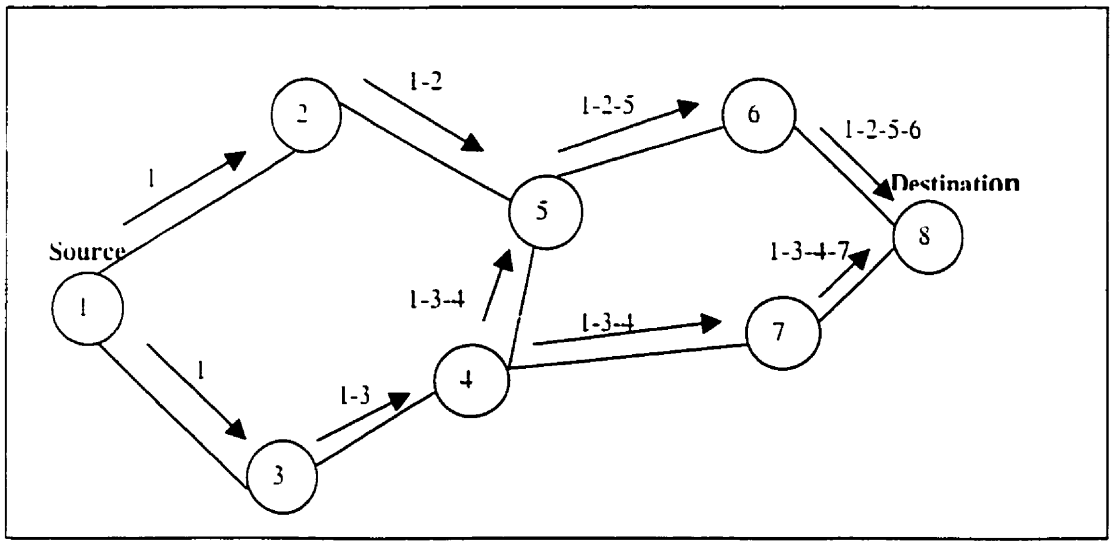

Figure 2.4 Building route records during route discovery in DSR

In order to return the route *reply* packet to the initiator of the route discovery, the target node must have a route to the initiator. If the responding node is an intermediate node, it will append its cached route to the route record and then generate the route reply. To return the route reply packet, the responding node must have a route to the initiator. If it has a route to the initiator in its route cache, it may use that route. Otherwise, the responding node may reverse the route in the route record from the route request packet, and use this route to send reply message to the initiator (See Figure 2.5).

Figure 2.5 demonstrates a scenario where the destination, node 8 replies with a reply packet upon the receipt of a request packet from node 1. Node 8 receives node 6's copy

before node 7's. As a result, it appends its own address to the route record of the request packet and includes a copy the reversed route record in the reply packet. Afterwards, node 8 forwards this reply packet to node 6 along the reverse path.

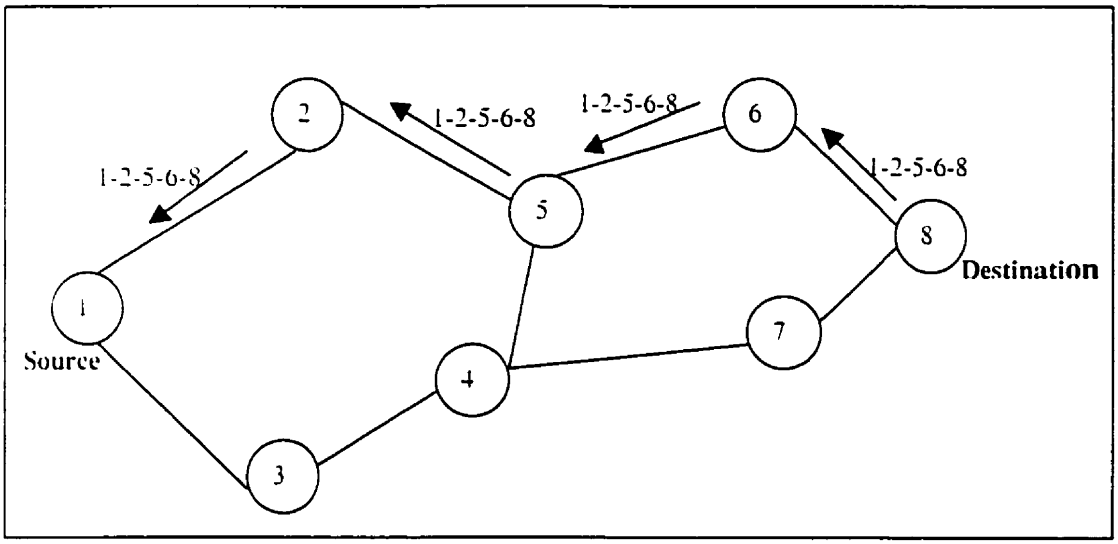

Figure 2.5 Forward back the reply message with route record in DSR

Route maintenance is accomplished through the use of route *error* packets and acknowledgments. When a route error packet is received, the hop in error is removed from the node's route cache and all routes containing the hop are truncated at that point. If this node has an entry for the original sender in its route cache, it may send the route error packet using that route. The source, upon the receipt of the error packet, restarts the route discovery process.

DSR uses source routing to avoid the need for intermediate nodes to maintain up-to-date routing information. Once a route has been discovered, there is no requirement for intermediate node routing tables. Routing information is contained entirely in the source routing header as a sequence of nodes over which the packet is to be forwarded. In the event that a next-hop along a source route is no longer reachable due to node mobility, a route failure message is sent back to the source, which must initiate a new route discovery process. The main benefit of DSR is that intermediate nodes do not need to respond at all to link failures unless a source directs them to--no routing information needs to be maintained at the intermediate nodes, thereby saving bandwidth and reducing power consumption compared with table-driven routing protocols. Also, cache routing saves overhead request packets generate. However, DSR does not adopt any mechanism to expire stale routes in the cache. If the stale routes are used, they would start polluting other caches, and later lots of packets would be dropped halfway. Furthermore, response to link failures may be slow because the intermediate nodes may not know they should be maintaining the route. Consequently, the source must initiate the route maintenance procedure. This could lead to excessive delays in the network.

2.2.2 Ad Hoc On-Demand Distance Vector (AODV)

Another on-demand routing protocol designed for ad hoc networks is Ad hoc On-demand Distance Vector (AODV) [9], which builds upon the DSDV algorithm previously

described. AODV is an improvement on DSDV because it typically minimizes the number of required broadcasts by creating routes on a demand basis, as opposed to maintaining a complete list of routes as in the DSDV algorithm. The authors of AODV classify it as a pure on-demand route acquisition system, since nodes that are not on a selected path do not maintain routing information or participate in routing table exchanges.

The path discovery process is initiated whenever a source node needs to communicate with another node for which it has no routing information in its table. The source broadcasts a route request (RREQ) packet to its neighbors. The RREQ packet is broadcast until the route to the destination is located. AODV utilizes destination sequence numbers to ensure all routes are loop-free and contain the most recent route information. Each node maintains its own sequence number, as well as a broadcast ID. The broadcast ID is incremented for every RREQ the node initiates, and together with the node's IP address, uniquely identifies an RREQ. Along with its own sequence number and the broadcast ID, the source node includes in the RREQ the most recent sequence number it has for the destination. Intermediate nodes can reply to the RREQ only if they have a route to the destination whose corresponding destination sequence number is greater than or equal to that contained in the RREQ. As the RREQ travels from a source to various destinations, it automatically sets up the reverse path from all nodes back to the source, as illustrated in Figure 2.6. To set up a reverse path, a node records the address of the

neighbor from which it received the first copy of the RREQ. These reverse path route entries are maintained for at least enough time for the RREQ to traverse the network and produce a reply to the sender.

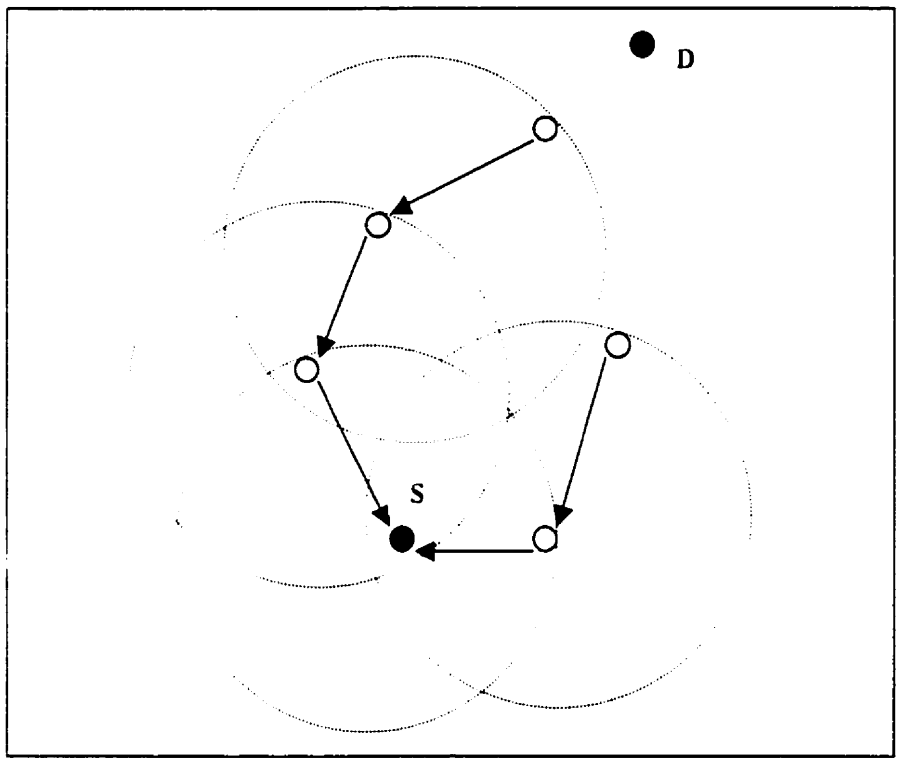

Figure 2.6 Reverse path formation in AODV

Once the RREQ reaches the destination, the destination responds by sending a route reply (RREP) packet back to the neighbor from which it first received the RREQ (see Figure 2.7). As the RREP is routed back along the reverse path, nodes along this path set up forward route entries in their route tables, which point to the node from which the RREP came. These forward route entries indicate the active forward route. 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.

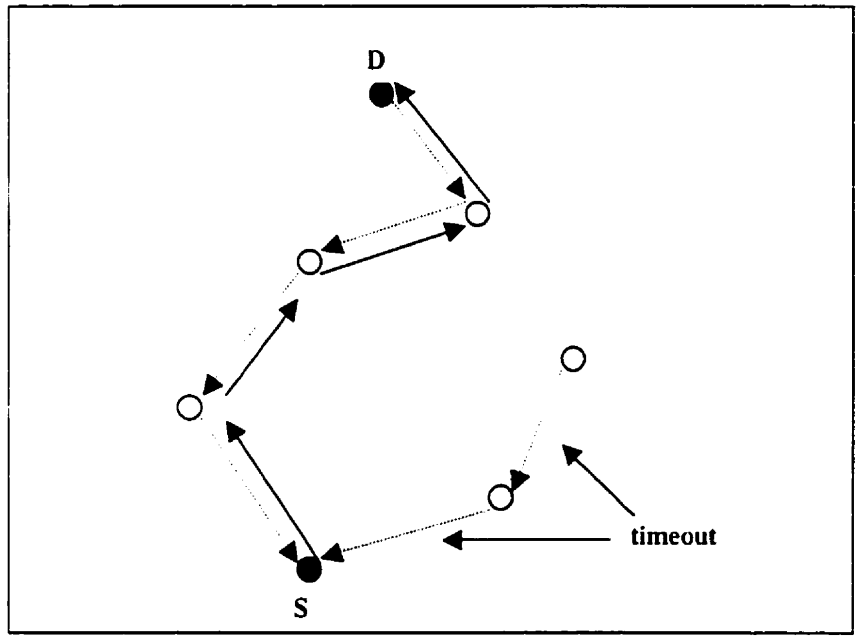

Figure 2.7 Forward path formation

Movement of nodes not lying along an active path does not affect the routing to that path's destination. If the source node moves during an active session, it can reinitiate the route discovery procedure to establish a new route to the destination. If a node along the route moves, its upstream[1] neighbor notices the move and propagates a link failure notification message (an RREP with infinite metric) to each of its active upstream

[1] If two nodes, i and j, are along a route from source to destination, then i is *upstream* of j if it relays packets to j in the direction of the destination and j is said to be *downstream* of i.

neighbors to inform them of the erasure of that part of the route. These nodes in turn propagate the *link failure notification* to their upstream neighbors, and so on until the source node is reached. The source node may then choose to reinitiate route discovery for that destination if a route is still desired.

An additional aspect of the protocol is the use of *hello* messages, periodic local broadcasts by a node to inform each mobile node in its neighborhood. Hello messages can be used to maintain the local connectivity of a node. If the hello message is not received within *hello_interval* period, path maintenance may be initiated.

2.2.3 Associativity Based Routing (ABR)

Node mobility plays a fundamental role in affecting the performance of ad-hoc network routing protocols. A protocol that provides superior performance under a given set of mobility patterns, may fail entirely under another. Consequently, if some notion of node mobility or it impact on network paths can be characterized, it should be possible to use this information to select more stable paths. Given the objective of minimizing routing algorithm reaction to node mobility, a protocol capable of routing along paths that experience the fewest link failures due to node mobility will be expected to outperform those that cannot. Consequently, the idea of stability as criteria for choosing paths has recently been advocated and incorporated into basic reactive routing protocols.

Associativity Based Routing (ABR) [5][6] represents the first attempt to factor node mobility into the routing process by proposing a model based on the concept of node associativity. ABR builds routes on a demand-basis using basic techniques that are similar to those used by DSR and AODV. Specifically, routes are constructed using a controlled flooding process, which effectively searches the network for a stable route toward a desired destination. The novel aspect of ABR is that it attempts to select routes that are long-lived, i.e., routes that are expected to survive longer than other routes. The objective is to reduce overall routing algorithm overhead by limiting the need to invoke route maintenance, which is normally required in response to node mobility. To achieve this, a new routing metric is proposed based on the concept of associativity--a measure of the duration of time that a radio link has been active between a pair of nodes. The argument is based on limited observations that nodes in an ad-hoc network will either rapidly move past each other, or will spend some dormant time, during which the relative movement of a pair of nodes is minimal, and the link remains stable. A predefined threshold is used to determine whether or not two nodes are in this dormant state, and hence exhibit high degree association stability.

Despite the novel approach advocated in ABR, there are serious shortcomings related to the associativity metric and the optimal path selection algorithm. Although the objective of the metric is to reflect how node mobility impacts link stability, it is not based on a well-defined model for node mobility. Instead, the metric relies entirely upon past

behaviors. Since node mobility and link characteristics are dynamic processes, the metric as defined is not a true predictor of future stability. It is merely a measure of past stability. Consequently, there is no quantitative basis for assessing the true stability of paths selected on the basis of this metric. Furthermore, the stability implied by longer associativity grows without bound. It cannot reflect that after some dormant time a mobile node is likely to eventually move relative to its neighbors for the reason that past association stability does not truly mean a link will survive forever. The best link available for routing to a rapidly moving node may not meet the associativity criteria which may operate well under less dynamic conditions.

2.3 Hybrid Routing Protocols

The idea of hybrid ad-hoc routing arises in most recent years, which combines table-driven routing and on-demand routing.

2.3.1 Zone Routing Protocol (ZRP)

Zone Routing [19] is another routing protocol designed for the ad hoc environment. It is a hybrid of on-demand routing with another table-driven routing protocol. In zone routing, each node defines its own zone as the nodes within certain distance of itself. Two different routing schemes are required for zone routing. For routing inside the zone, any

routing schemes, including Distributed Bellman-Ford or Link State routing [20], can be used. The goal for this intra-zone routing is to maintain a full information table about the reachability of nodes within a region. For the inter-zone routing, it uses the on-demand routing to find the path. Combining these two routing schemes, zone routing operates like this: when there is a packet that needs to be routed, a node checks whether the destination is within the zone. If it is, since the intra-zone routing scheme maintains the necessary information, it can be routed directly. When it comes to route traffic to a destination outside a node's zone, zone routing searches for the path by multicasting request packets to the border nodes, using the shortest paths provided by the intra-zone protocol. If some border nodes know of a route to this destination, the response packets will then be sent back to the source. Otherwise, the border nodes keep requesting their border nodes, in the same fashion, for a route to the destination.

The advantage of zone routing is its scalability, as it reduces the need for a large storage for the routing table. But since it has a component resembling table-driven routing, it has the same problem of considerable signaling traffic and power consumption.

2.3.2 Virtual Base Stations Protocol (VBS)

Virtual Base Stations (VBS) Protocol [23] is neither a purely table-driven nor a purely on- demand routing protocol. This protocol is built upon the VBS infrastructure-creation

protocol. In VBS, some of the mobile nodes, based on an agreed-upon policy, become in charge of all the nodes in their neighborhood, or a subset of them, which is achieved by electing one to be a virtual base station. Unlike table-driven routing protocols, not all mobile nodes are required to have complete knowledge of the network, and unlike on-demand routing protocols, routes between nodes of the ad-hoc network are not built only based on request packets from source nodes. In VBS routing, a mobile node wishing to send a packet to another mobile node in the network, sends the packet to its VBS, which forwards it to the destination itself, the VBS in charge of the destination, or to the correct border mobile node, based on the information stored in its VBS table.

The advantage of VBS is its scalability to networks with large populations of mobile nodes. However, as mobility increases, frequent VBS changes can adversely affect routing protocol performance since nodes are busy in virtual base station election rather than packet relaying.

2.4 Summary

In this chapter, we have provided descriptions of several routing schemes proposed for ad hoc mobile networks. We have also provided a classification of these schemes according to the routing strategy, i.e., table-driven, on-demand or hybrid routing. Table-driven ad hoc routing approaches rely on an underlying routing table update mechanism that

involves the constant propagation of routing information. This is not the case, however, for on-demand routing protocols. When a node using an on-demand routing protocol desires a route to a new destination, it will have to wait until such a route can be discovered. On the other hand, because routing information is constantly propagated and maintained in table-driven routing protocols, a route to every other node in the ad hoc network is always available, regardless of whether or not it is needed. This feature, although useful for datagram traffic, incurs substantial signaling traffic and power consumption, yielding inferior performance to on-demand routing.

Among all proposed wireless mobile ad-hoc routing protocols, Dynamic Source Routing (DSR) [7][8] and Ad-hoc On-demand Distance Vector (AODV) [9] are the most prominent, nominated by the Internet Engineering Task Force (IETF) Mobile Ad-hoc NETworking (MANET) working group [22] as candidates for standardization. It has been long believed that the performance of ad hoc networks routing protocols is enhanced when nodal mobility is reduced. This is true when considering performance measures such as packet delivery fraction and routing overhead. This may not be the case, however, when we consider packet delay. It was shown in [21] that the packet delay for both AODV and DSR increases as the nodal mobility is reduced. This is because there is a tendency in ad hoc networks routing protocols to use a few "centrally located" nodes in a large number of routes. This causes congestion at the medium access control (MAC) level, which in turn may lead to high packet delays, since few nodes have to carry

excessive loads. Moreover, such nodes would also suffer from high battery power consumption. This is an undesirable effect, which is compounded by the limited battery power of the mobile terminals. In fact, a major drawback of all existing ad hoc routing protocols is that they do not have provisions for conveying the load and/or quality of a path during route setup. Hence they cannot balance the load on the different routes. In Chapter 3, we introduce a new routing protocol, based on the concept of balancing traffic load, which achieves better performance than both DSR and AODV.

Chapter 3

LBAR Routing

In this chapter, we present Load-Balanced Ad-hoc Routing (LBAR), which uses nodal activity to effectively balance traffic load among the different nodes in wireless ad-hoc networks. Section 3.1 is an overview of LBAR routing supplemented with examples to ease the explanation. Section 3.2 contains a detailed description of the algorithms executed by the mobile nodes running the LBAR routing protocol. Finally, Section 3.3 is a chapter summary.

3.1 Overview of the scheme

The proposed Load-Balanced Ad-hoc Routing (LBAR) is an on-demand routing protocol intended for delay-sensitive applications where users are most concerned with packet transmission delay. Hence, LBAR focuses on how to find a path, which would reflect least traffic load so that data packets can be routed with least delay. The algorithm has

four components: *Route Discovery* (Section 3.1.1); *Path Maintenance* (Section 3.1.2); *Local Connectivity Management* (Section 3.1.3); *Cost Function Computation* (Section 3.1.4).

LBAR uses a broadcast *route discovery* mechanism (Section 3.1.1), which allows any node in the ad hoc network to dynamically discover a route to any other node in the ad hoc network, whether directly reachable within wireless transmission range or reachable through one or more intermediate network hops through other nodes. When one source node wants to send packets to a targeted node, it first starts the route discovery protocol. A setup message is propagated to its neighbors at which route cost (described in Section 3.1.4) carried in the message is updated based on activity value of the node. When this request packet arrives at the destination, a route with its cost is recorded in the routing table of destination. The procedure of choosing a least-load-cost route is initiated by the destination before the expiration of a route-select timer.

One distinct feature of ad hoc networks is node mobility. For example, if the sender, the destination, or any of the other nodes along a route move out of the wireless transmission range of the next or previous hop along the route, the route can no longer be used to reach the destination. As well, a route will no longer be valid if any of the nodes along the route should fail or are powered off. When local connectivity management (Section 3.1.3) detects a problem with a route in use, a process is initiated to correct the route to the

destination, or in the worst case, to inform source to restart route discovery. This monitoring of the correct operation of a route in use is known as *path maintenance*.

3.1.1 Route Discovery

The route discovery process is initiated whenever a source node needs to communicate with another node for which it does not have a known route. The process is divided into two stages: *Forward* and *Backward*. The forward stage starts at the source node by broadcasting setup messages to its neighbors. A setup message carries the cost seen from the source to the current node. A node that receives a setup message will forward it, in the same manner, to its neighbors after updating the cost based on its nodal activity value. Otherwise, if the node does not have any neighbors in the communication range to relay, the setup message is simply discarded. The destination node collects arriving setup messages within a waiting period, which is a predefined timer for selecting the best-cost path.

The backward stage begins with an ACK message forwarded backward towards the source node along the selected path, which we call the *active path*. If a link on the selected path breaks, the ACK message is discarded and an error message is sent backward along the path fragment to the destination. The destination node will then choose another path, which does not contain any of the previous broken links. When the

source node receives an ACK message, it knows that a path has been established to the destination and then starts transmission. When transmission is completed, the destination removes all the corresponding stale routing information from its table.

In order to prevent looping when setup messages are routed, all setup messages are assumed to contain a route record, including a list of all node IDs used in establishing the path fragment form the source node to the current intermediate node. In addition, all ACK messages are assumed to contain a list of broken links encountered in trying to set up the corresponding connection. These lists must be checked when rerouting so that loops and previous broken links can be avoided.

3.1.2 Path Maintenance

In wireless networks, nodes are allowed to move freely, which causes dynamic topology changes and route invalidity. Movement of nodes not lying along an active path does not affect the routing to that path's destination. However, if the source node, an intermediate node on the active path or the destination node moves out of the communication range, an alternate path must be found.

If the source node moves away from the active path during an active session, packets would not be able to be relayed to downstream neighbors. In this case, the source has to

reinitiate the route discovery procedure to establish a new route to the destination. Figure 3.1 illustrates a case of the source node mobility. The source node S moves from [a] to [b] and then the virtual link between S and 3 is broken. So path S–3-4-5-6-7-8-DEST is on longer valid. Node S has to restart route discovery procedure to find a new route to the destination node DEST, say S–1-2-6-7-8-DEST.

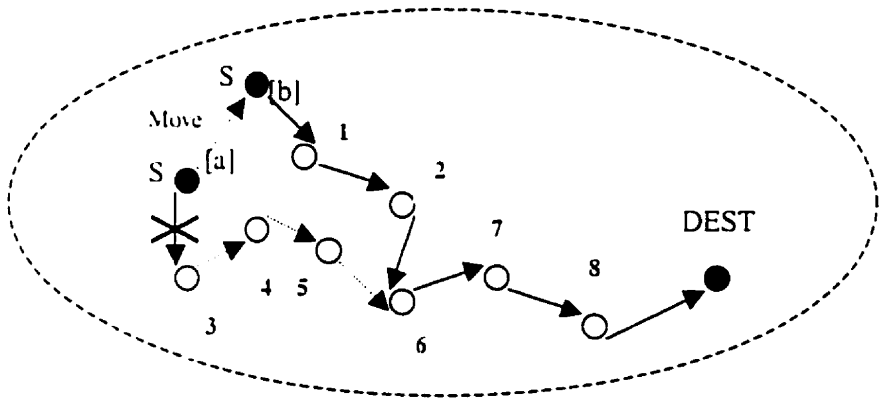

Figure 3.1 Source mobility outside active path

When either the destination node or some intermediate node moves outside the active path during a session, path maintenance will be initiated to correct the broken path. Periodic hello messages can be used to detect link failures, as will be described in Section 3.1.3. A link failure is also indicated if attempts to forward a packet to the next hop fail. Once the next hop becomes unreachable, the node upstream of the broken hop propagates an error message with sender ID to the destination. Those nodes subsequently relay that message to their neighbors and so on. This process continues until the destination is

notified. Upon receiving notification of a broken link, the destination node picks up an alternative best-cost partial route passing through the node propagating the error message and then sends an ACK message to the initiator of the error message. If the destination has no alternative path passing through the node sending the error message, the destination picks up another route and sends an ACK message to the source. The source will use this new route to send data packets if it still has data to send. By then, a new active path is defined. In the worst case, where the destination has no alternate paths, it propagates an error message to the source and lets it restart route discovery.

The example in Figures 3.2 and 3.3 illustrates a scenario, where an intermediate node moves away from the active path. The scenario shows intermediate node 4 moving from [a] to [b] (not on active path to the destination DEST), hence, path S-1-2-3-4-5-DEST becomes inaccessible. Node 3 cannot hear node 4, so it broadcasts an error message. When DEST receives the error message, there are three cases:

- DEST has a path going through the error initiator node – node 3. DEST sends an ACK to node 3, which is responsible for redirecting data packets to DEST along S-1-2-3-6-7-DEST (see Figure 3.2).
- DEST has no path through node 3. DEST then chooses a path with best cost, like S-1-2-6-7-DEST, and then sends ACK to source S (see Figure 3.3).
- DEST has no alternate path and floods an error message to the source S.

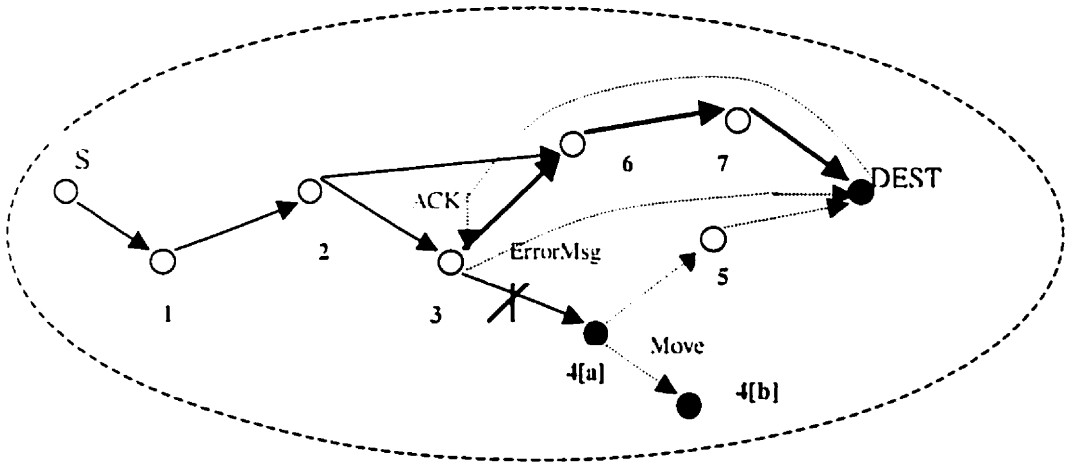

Figure 3.2 Destination/intermediate node has alternate path passing through error initiating node

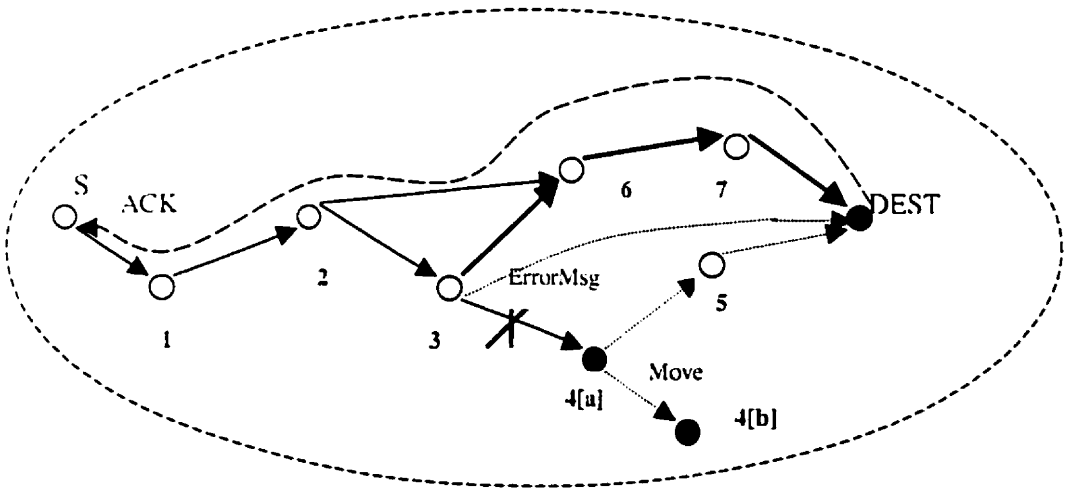

Figure 3.3 Destination/intermediate node has alternate path not passing through error initiating node

3.1.3 Local Connectivity Management

Nodes learn of their neighbors in one of two ways. Whenever a node receives a broadcast from a neighbor, it updates its local connectivity information in its *Neighborhood table* to ensure that it includes this neighbor. In the event that a node has not sent data packets to any of its active neighbors within a predefined timeout, *hello_interval*, it broadcasts to its neighbors a *hello* message, containing its identity and activity (described in Section 3.1.4). This hello message is prevented from being rebroadcast outside the neighborhood of the node. Neighbors that receive this packet update their local connectivity information in their Neighborhood tables. Receiving a broadcast or a hello from a new neighbor, or failing to receive consecutive hello messages from a node previously in the neighborhood, is an indication that the local connectivity has changed. Failing to receive hello messages from inactive neighbors does not trigger any protocol action. If hello messages are not received from the next hop along an active path, the upstream active neighbors using that next hop send notification of link failure and the path maintenance protocol is started.

3.1.4 Cost Function Computation

The cost function is used to find a path with the least traffic so that data packets can be transmitted to the destination as fast as possible while achieving the goal of balancing load over the network. The following definitions are used:

- *Active path*: a path from a source to a destination, which is followed by packets along this selected route.
- *Active node*: a node is considered *active* if it originates or relays data packets or is a destination.
- *Inactive node*: a node is considered inactive if it is not along an active path.
- *Activity*: The number of active paths through a node is defined as a metric measuring the activity of the node. The more active a node is, the more load the node is burdened with.
- *Cost*: Minimum traffic interference (defined below) is proposed as the metric for best cost.

In wireless ad hoc networks, nodes use radio signals for communication. Communication among mobile nodes is limited within a certain transmission range. Within each such range, only one transmission channel is used, covering the entire available bandwidth. To transmit data, mobiles within the same range have to sense for other transmissions first, then gain an access permit, and transmit only if no other node is currently transmitting. Unlike wired networks, packet delay is not caused only from traffic load at the current node, but also by traffic load at neighboring nodes. We call this *traffic interference*. In the context of traffic interference, the best-cost route is regarded as a path which encounters the minimum traffic load in transmission and minimum interference by neighboring nodes. To assess best cost, the *node activity* metric is used as an indirect means to reflect traffic load at the node. Such activity information can be gained at the network layer,

independent of the MAC layer. Traffic interference is defined as the sum of neighboring activity of the current node. During the routing stage, nodal activity and traffic interference are calculated at every intermediate node along the path from source to destination. When the destination receives routing information, it chooses a path which has minimum cost.

- Activity A_i: Number of active paths through node i. The greater the value of activity is, the more traffic passing through node i would be.
- Traffic interference TI_i: $TI_i = \sum_{j \in neighbourhood(i)} A_{i,j}$, which is the sum of activity of neighboring nodes of node i, where j is a neighboring node of node i and $A_{i,j}$ is the activity of neighboring node j of node i.
- Cost C_k: cost of route k.

$$C_k = \sum_{i \in k} (A_i + TI_i) = \sum_{i \in k} (A_i + \sum_{j \in neighbourhood(i)} A_{i,j})$$

where i is a node on path k other than source and destination and j is a neighboring node of node i. (Every path with identified source-destination pair includes the same source and destination, so for simplicity, activities of source and destination are excluded.)

This is a generic cost function, which is based on the assumption that packets are of the same size and traffic is at a constant rate. Interference from neighbors must be considered to be a factor of the function, in addition to current nodal activity. This is because mobile

nodes within the same range compete for media access, which causes packets to be delayed for transmission.

As Figure 3.4 presents, mobile node S, a and b are within the range of each other. If S would like to send a packet to node a, S has to sense if node a and b are transmitting. If any of these three nodes is transmitting, the packet at node S has to back off for some time. Otherwise, node S can transmit the packet. Therefore, packet delay at a mobile node is not only attributed to load at the current node, but to traffic from neighboring nodes as well. Hence, our cost function is a combination of activity of nodes along the path and activity of their neighboring nodes.

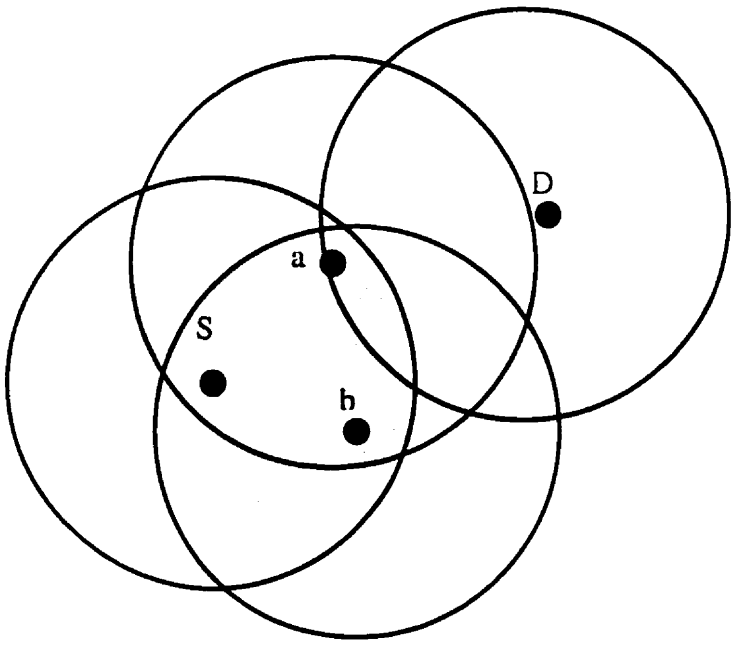

Figure 3.4 Example of traffic interference

Figure 3.5 exemplifies a scenario of how to calculate cost and to select the best cost route. Source node 1 broadcasts a setup message to its neighbors, node 2 and node 6, until the message reaches destination node 5. Suppose that one active path passes through node 1, three active paths pass through node 2, two through node 3, two through node 4, three through node 5, two through node 6, and one through node 7. So the activity of node 1 is 1. Similarly, for nodes 2, 3, 4, 5, 6 and 7, the activity is 2, 2, 3, 2, and 1, respectively.

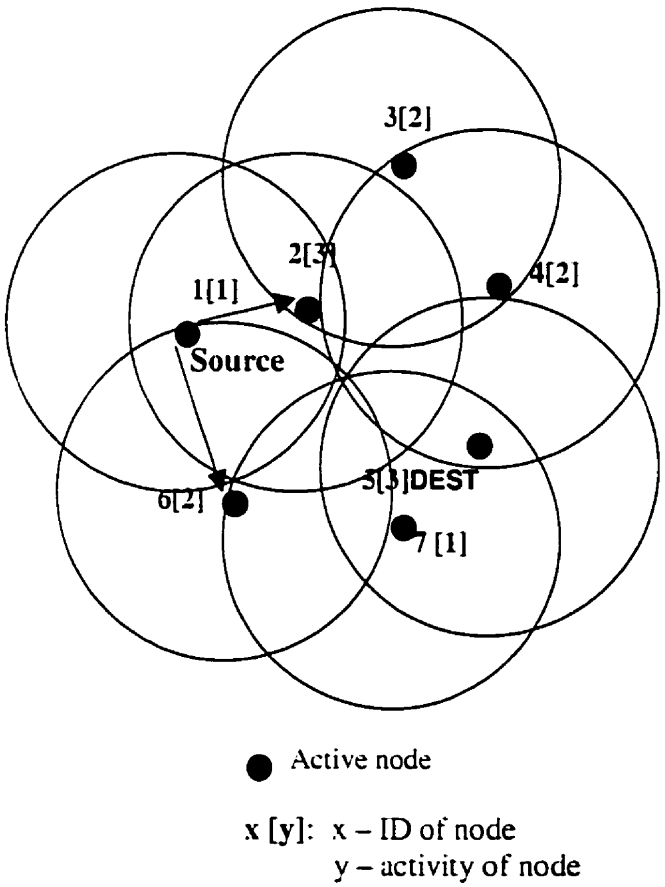

Figure 3.5 Broadcast setup message to neighbors from source

We construct a virtual wired network (see Figure 3.6) by drawing a link if two nodes can hear each other. From this graph, four routes from source node 1 to destination node 5 are found: 1-2-3-5, 1-2-4-5, 1-2-3-4-5, 1-2-4-3-5 and 1-6-7-5. Consider the path 1-2-3-5 for example. Node 2 has its own activity of 3. Its neighbors are node 1, node 3 and node 4 with a total traffic interference value of 5 to node 2. So the cost of passing through node 2 is 8. For node 3, cost is 10. Since node 1 is the source and node 5 is the destination, the cost of passing these two nodes is not included. Hence, the cost of this path is 18. Similarly, for route 1-2-4-5, 1-2-3-4-5, 1-2-4-3-5, 1-6-7-5, the cost is 18, 28, 28, and 10, respectively. Therefore, 1-6-7-5 is selected as a minimum-cost path reflecting relatively least traffic.

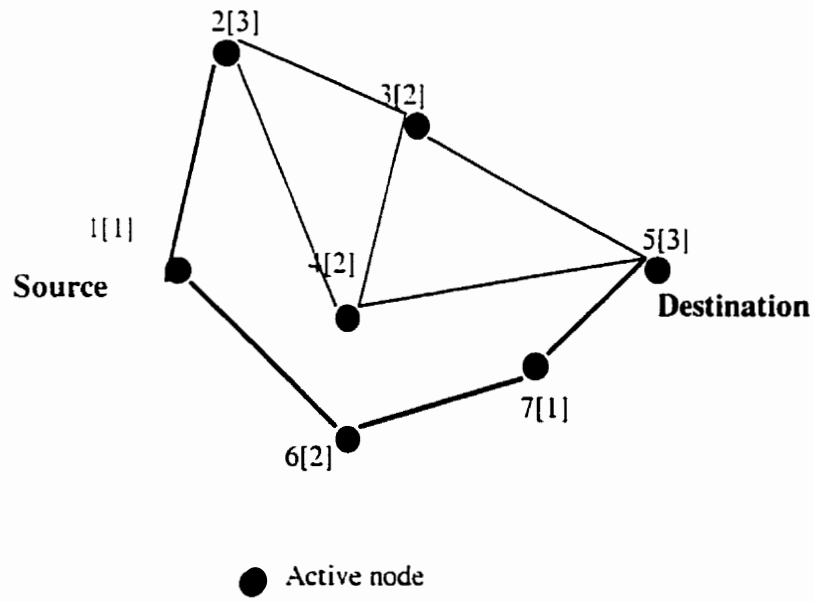

Figure 3.6 Mapping Figure 3.5 to a virtual wired network

3.2 Detailed Description of LBAR Routing

This section provides a detailed description of the LBAR scheme. This includes messages exchanged, tables and variables used by the scheme, in addition to pseudo code description for each node participating in the algorithm. These are the source nodes, the destination nodes and intermediate nodes.

3.2.1 Messages Used by the Scheme

SetupMsg (source_addr, broadcast_id, dest_addr, IDs[1]*, cost)*

A route request message with the following parameters:

- *Source_addr*: active source address
- *Broadcast_id*: a counter which is incremented whenever the source sends a SetupMsg
- *Dest_addr*: destination address
- *IDs*: route list - record nodes IDs visited by SetupMsg
- *Cost*: the cost value of the path from the source to the current node. The destination node chooses the path with minimum cost

ACK (source_addr, broadcast_id, dest_addr, ACK_dest, IDs, path)

[1] The IP address of a node can be used as its ID.

An acknowledgement message which contains the same parameters of SetupMsg, except cost. In addition, it contains the confirmed *path*, which represents a list of links from destination node to the current node, and *ACK_dest*, which indicates the node the ACK message is intended for.

ErrorMsg (source_addr, broadcast_id, dest_addr, error_sender, path, break_links)

An error message used to inform the destination to pick an alternate path whenever a broken link is observed. Its parameters are the same as the ACK message, except ACK_dest. In addition, it contains *error_sender*, which is the address of the node detecting the link break and sending the error message.

Hello (hello_addr, activity)

A message used to guarantee local connectivity. It contains two parameters: the identity of the node sending Hello message, and an indication of nodal activity.

3.2.2 Local Nodal Information

A number of tables are kept at different nodes for use in routing packets and path maintenance. These are:

Source Routing table: Records route to destination and includes the following information:

- broadcast_id: a counter which is incremented whenever the source sends a *SetupMsg*
- dest_addr: the destination address
- path: the path from source to destination

Active Path table: Records information of active paths at intermediate nodes and includes the following information:

- source_addr of active source
- broadcast_id
- dest_addr
- next hop: downstream node ID on the active path (only if downstream node does not inform its existence, will current node initiate path maintenance)
- new path: record of alternate path to redirect data packets
- timer: purge this entry when expiring

Neighborhood table: An intermediate node maintains IDs and activity of its neighbor nodes, including:

- neighbor_id: the ID of neighbor
- activity: activity of neighbor

Destination Routing table: The destination maintains a list of all possible paths, including the following information:

- source_addr of active source
- broadcast_id
- path
- cost: the cost of an identified path
- timer: this entry is out of date when expiring

3.2.3 The Algorithm for the Source Node

A pseudo code description of the algorithm for the source node is shown in Figure 3.7. Lines 1-4 represent the beginning of the forward stage of the scheme, where a request to establish a path is initiated. The source node begins to forward SetupMsg to its neighbors. Lines 5-7 indicate the path has been found, which is contained in the ACK message arrival. Therefore, the source can begin transmitting data after the routing information is written in the routing table. When transmission is completed, routing information will not be removed until *source_route_timer* expires, which is represented by lines 8-10. Lines 11-12 describe the case that source restarts the request if it does not receive ACK until route discovery timer expires. When source receives an ErrorMsg indicating that destination cannot find alternate paths, it also restarts route discovery, which is handled by lines 13-14.

3.2.4 The Algorithm for Intermediate Nodes

A pseudo code description of the algorithm for an intermediate node in any reachable path is shown in Figure 3.8. Lines 1-3 represent the forward stage of the scheme, where the node forwards SetupMsg to its neighbors, avoiding already visited nodes. Lines 4-7 show the backward stage of the algorithm, in which ACK is forwarded upstream if next link is not broken. Otherwise, ErrorMsg is sent from the node along the path fragment to indicate the failure of the candidate path. When an ErrorMsg is received, the message is

relayed downstream until it reaches the destination to pick an alternative path, which is represented by lines 8-13. To guarantee that data packets arrive at the intended destination, packets must be redirected to a new partial route; see lines 14-25. Lines 26-27 indicate that stale information kept in the active path table is removed, which means this node is no longer on this active path.

3.2.5 The Algorithm for the Destination Node

A pseudo code description of the algorithm for the destination node is shown in Figure 3.9. Lines 1-8 represent the forward stage and the start of the backward stage. The information carried by the setup message is stored at the destination routing table. If the route-select time period enforced at the destination node is reached, the path with the minimum cost is selected to begin the backward stage. This is performed only if no *ACK* message has been already released from the destination. Lines 9-19 represent the case when receiving ErrorMsg, where the destination is informed of a broken link on the path and then removes all the invalid paths associated with that broken link. Another path, which does not contain any previously broken links, may be selected to restart the backward phase. Lines 20-21 indicate that the destination removes stale routing information from routing table after route timer expires in the case that data transmission has been terminated.

```
For source node S

1.  If setup message is ready{
2.     Broadcast SetupMsg (source_addr, broadcast_id, dest_addr, IDs, cost)
       // Forward the Setup Message to all available neighbor nodes of S
       {
3.       ∀node ∈ Neighbors of S
4.       send SetupMsg (source_addr, broadcast_id, dest_addr,
                              IDs + S, 0)
        }

5.  Receive ACK (source_addr, broadcast_id, dest_addr, ACK_dest, IDs, path)
    from link L
    // Acknowledgement is sent back from destination to source S
    {
6.       Write routing information in the routing table
                ( broadcast_id, dest_addr, path)
7.       Start transmission
    }

8.  When data transmission is completed
9.   If source_route_timer expires {
10.    Remove the entry in routing table of the source S
                ( broadcast_id, dest_addr, path )
     }

11. If source S does not receive ACK in δ (call waiting period){
12.   Go to Line 1.}


13. Receive ErrorMsg (source_addr, broadcast_id, dest_addr, error_sender, null, null)
    // Route error message received from destination
    {
14.   Go to Line 1.
    }
```

Figure 3.7 The algorithm for source node

```
For intermediate node I

1.  Receive SetupMsg (source_addr, broadcast_id, dest_addr, IDs, cost)
      // Forward the Setup Message to all available neighbor nodes of Node I
    {
2.          ∀node ∈ Neighbor of s and node ∉ IDs
3.          Send SetupMsg (source_addr, broadcast_id, dest_addr, IDs + I, cost +
                                Cost(I) )
    }


4.  Receive ACK (source_addr, broadcast_id, dest_addr, ACK_dest, IDs, path)
    {
5.      If next (IDs) is not broken
          // If ACK_dest != source_addr, which means ACK contains a new path
          // to acknowledge the node detecting link breakage, Send ACK to
          // ACK_dest; Else which means ACK is to acknowledge source node
          // Send ACK to source_addr
        {
          Send ACK (source_addr, broadcast_id, dest_addr, ACK_dest, IDs,
                        path + I) to next (IDs)
          Build active path:
            Record necessary information in the active path table of node I
              (source_addr, broadcast_id, dest_addr, First_Element (path) )
        }
6.      Else
7.          Send ErrorMsg (source_addr, broadcast_id, dest_addr, null, path,
                                break_links + L) to next (path) to destination
    }


8.  Receive ErrorMsg (source_addr, broadcast_id, dest_addr, error_sender, path,
                        break_links) from link L
    // Inform the destination to pick up another path
    {
9.      If path! = null
        {
10.       Send ErrorMsg to next (path)       //along reverse path
11.       Remove corresponding stale information from active path table
            (source_addr, broadcast_id, dest_addr, next_hop)
        }
```

```
12.      Else  // notify destination to choose a new route
13.          Flood ErrorMsg to neighbors
     }

Data transmission
14. If active path is broken and new path == null
    {
15.    If Error message has not yet been sent to destination
       {
16.       Propagate ErrorMsg until it reaches destination
17.       Buffer this data packet
       }
18.    else  // Error message has been sent to destination
       {
19.       Buffer this data packet

       }
    }
20. If active path is broken and new path != null
    // new path is stored in the active path table
    {
21.    Direct this packet along new partial route to destination
22.    Reset timer in the active path table
    }
23. If active path is still valid
    {
24.    Send data packet to next hop
25.    Reset timer in the active path table
     }


26. If the timer in the active path table expires {
        // This node has not received data packets during a predefined time limit
27.    Remove stale information kept in the active path table}
```

Figure 3.8 The algorithm for intermediate node

```
For Destination

1.  Receive SetupMsg (source_addr, broadcast_id, dest_addr, IDs, cost) from Link L
    {
2.      Store the contained information into the destination routing table
3.      If Waiting time period τ is exhausted {
4.          Select path with best cost -- Min (cost)
5.          Send ACK (source_addr, broadcast_id, dest_addr, ACK_dest,
                        IDs, null) to next (IDs) }
6.      Else {
7.          Wait until τ is reached
8.          Go to Line 3}
    }

9.  Receive ErrorMsg (source_addr, broadcast_id, dest_addr, error_sender, path,
                        break_links) from link L
      // Remove paths that contain the broken links and select alternate path
    {
10.     If ∃ path: break_links ∉ path and error_sender ∈ path {
11.         If error_sender != null
12.             Send ACK (source_addr, broadcast_id, dest_addr,
                        ACK_dest, IDs, null) to next (ID s) to error_sender
13.         Else
14.             Send ACK to source}
15.     Else
16.         choose a new path with best cost
17.     If destination can not find an alternative path {
18.         Remove corresponding routing information from destination routing table
                // Worst case, source has to restart routing.
19.         Send ErrorMsg to source}
    }

20. If route timer expires{
        // Transmission of Data packets has been terminated
21.     Remove stale routing information kept in the routing table}
```

Figure 3.9 The algorithm for destination node

3.3 Summary

In this chapter, we provided a detailed description of the operation of the LBAR protocol. LBAR is a distributed, destination-controlled, load-balancing routing protocol that is intended for dynamic ad-hoc networks. Illustrative examples of LBAR were presented in Section 3.1. A detailed description and pseudo code of the algorithms executed by nodes running LBAR were provided in Section 3.2.

It should be noted that although LBAR requires different functionality from various nodes in the network, its implementation is a straightforward one. We list some of the desirable characteristics of LBAR:

- LBAR is loop free, since control messages include a list of all nodes IDs used in establishing the path fragment form the source node to the current intermediate node. As a result, loops can be instantly detected.
- LBAR is completely distributed and does not require knowledge of the global network state. Nodes only use their local database.
- The most prominent attribute of LBAR is that LBAR tries to balance traffic by routing data along less congested paths to avoid heavily-loaded nodes. This achieves much lower average end-to-end delay (as will be shown in Chapter 4).
- The routing decision is destination controlled as opposed to the traditional source control that is usually exerted in routing algorithms. The destination collects

information of all possible paths in its routing table and then makes a selection of the path with the best-cost value.

- LBAR detours data packets along a new path to the destination in case of link failure as opposed to the traditional routing protocol, which drops packets and restart route discovery. This new characteristic delivers more packets to the destination. As a consequence, better performance is achievable in terms of packet delivery fraction.

Chapter 4

Performance Evaluation of LBAR Routing

In this chapter, the performance of LBAR routing is studied. The results are analyzed and compared with AODV [9] and DSR [8]. The reason to choose AODV and DSR among all the proposed ad-hoc routing algorithms in Chapter 2 is that they are the most eminent. Both algorithms have been nominated as candidates for standardization by IETF. Section 4.1 describes the simulation model developed for evaluating the performance of LBAR routing, which includes the traffic model and the mobility model. Section 4.2 describes the performance metrics that are taken into consideration, which include packet delivery fraction, average end-to-end delay and normalized routing load. The results of our investigation of the effect of traffic load, node density, and node mobility on the performance of the LBAR routing are reported in Section 4.3. Finally, Section 4.4 provides a summary of the results obtained in this chapter.

4.1 Simulation Model

To evaluate the performance of our LBAR protocol, we construct a packet-level simulator that allows us to observe and measure the protocol's performance under a variety of conditions. The model is similar to that in [21]. The parameter settings used in this chapter are in conformance with those in [21]. Note that parameter probing was extensively performed in [21] and that it was demonstrated in the paper that such parameter settings resemble a wide range of application and mobility requirements.

In addition to a number of parameter choices in the protocol, the simulator allows us to vary certain environmental factors such as the number of mobile nodes, the number of traffic sources, and speed of node movement. Our simulations are run using ad-hoc networks of 50 and 100 mobile nodes. This is actually done so that networks with different node densities are simulated. The simulated wireless mobile ad-hoc network had a nominal bit rate of 2 Mbps. A 1500m × 300m grid is used for 50-node case, and 2200m × 600m grid is used for 100-node case. The mobile nodes are distributed randomly in the closed coverage area. During the simulation, nodes are free to move anywhere within this area.

The interconnection pattern of an ad-hoc network is determined in part by the communication range (R_{max}). For our simulations, we hold R_{max} constant at 250m. Two

nodes can communicate directly, and are thus considered each other's neighbors, if they are less than R_{max} distance apart.

A CSMA technique with collision avoidance (CSMA/CA) is used to transmit packets [17] [18]. Before beginning a transmission, carrier sensing is performed by a node to determine whether any of its neighbors is transmitting. If the node detects an ongoing transmission by a neighbor, it backs off a predefined time and waits before listening to the channel again. A node attempts to transmit a packet max_retrans times before dropping the packet. The maximum number of allowed retransmissions for any packet is 3.

Our simulation model maintains a send buffer of 64 packets. It buffers all data packets waiting for a route, i.e., packets for which route discovery has started, but no reply has arrived yet. To prevent buffering of packets indefinitely, packets are dropped if they wait in the send buffer for more than 30 sec. All packets (both data and routing) sent by the network layer are queued at the interface queue until the MAC layer can transmit them. The interface queue is FIFO, with a maximum size of 64. Routing packets are given higher priority than data packets in the interface queue.

The routing protocol will be initiated whenever a node has packets to send or whenever it receives a packet from a neighbor. Mobile nodes broadcast their hello messages every 0.1

second to confirm its relationship with neighbors. Simulations are run for 900 simulated seconds for 50 nodes, and 500 simulated seconds for 100 nodes. The traffic and mobility models are described in Sections 4.1.1 and 4.1.2, respectively.

4.1.1 Traffic Model

Traffic sources are CBR (continuous bit rate). The source-destination pairs are spread randomly over the network. We diversify the number of traffic pairs to change the number of packets to be routed by the network. The more traffic sources, the more packets generated by the traffic sources. 10, 20, 30 and 40 traffic sources are used when the number of nodes is 50. On the other hand, 10, 20 and 40 traffic sources are used for 100-node case. We use a packet rate of 4 packets/second, except for 40 traffic sources for 100-node case which use 2 packets/second.

4.1.2 Mobility Model

Nodes are initially placed randomly within a fixed-size rectangular field. During the simulation, nodes are free to move anywhere within this area. Each node moves with a velocity from a uniform distribution between 0 and 20 meters per second. Each host is initially placed at a random position within a rectangular area. As the simulation progresses, each host pauses at its current location for a period, which we call the pause

time, and then randomly chooses a new location to move to. Each host continues this behavior, alternately pausing and moving to a new location for the duration of the simulation. This process repeats throughout the simulation, causing continuous changes in the topology of the underlying network. In addition, we vary the pause time, which reflects the relative speeds of the mobiles. The smaller the pause period, the higher the mobility while the greater the pause period, the lower the mobility. This implies that varying the length of the pause period is equivalent of varying the mobility model. Experiments are run for pause periods of 100, 300, 600, 900 seconds with 50 nodes, and 100, 300 and 500 with 100 nodes.

4.2 Performance Metrics

Three key performance metrics are evaluated. These are:

1. *Packet delivery fraction* – this is the ratio of the data packets delivered to the destination to those generated by the CBR sources. The greater the packet delivery fraction, the more reliable the routing protocol, and the less the probability of dropping a data packet.
2. *Average end-to-end delay* of data packets – this includes all possible delays caused by queuing for transmission at the node, buffering data for detouring, retransmission delays at the MAC, propagation delay and transmission time. The smaller the average end-to-end delay, the faster the transmission of data packets.

3. *Normalized routing load* – this measures the number of routing packets transmitted per data packet delivered at the destination. Each hop-wise transmission of a routing packet is counted as one transmission. The lower the normalized routing load, the less overhead the routing protocol produces.

The packet delivery fraction reflects the degree of reliability of the routing protocol. The average end-to-end delay represents the quality of the routing protocol. The normalized routing load metric evaluates the efficiency of the routing protocol in terms of extra load introduced to the network. Nevertheless, if the average end-to-end delay is small even though the packet delivery fraction is high and the normalized routing load is small, this definitely means that the routing protocol provides delay-sensitive, reliable, and low overhead communications.

4.3 Simulation Results

The experiments in this chapter use different number of sources with a moderate packet rate and changing pause times. A 90% confidence level, with 10% intervals, was used in the simulations, see appendix. The confidence intervals for LBAR were usually within 1% -3% of the mean value. For clarity of presentation, confidence intervals are not explicitly depicted on the performance plots.

For the 50 node experiments we use 10, 20, 30, and 40 traffic sources and a packet rate of 4 packets/sec. The results of the experiments for the 50-node case are shown in Figures 4.1, 4.2 and 4.3. Figure 4.1 shows the packet delivery fractions for variations of the pause time for LBAR, AODV, and DSR. Note that the packet delivery fractions for LBAR, AODV, and DSR are very similar for both 10 and 20 sources (see Figure 4.1 (a) and (b)). With 30 and 40 sources, however, LBAR outperforms AODV and DSR (see Figure 4.1 (c) and (d)). In fact, LBAR achieves the highest packet delivery fraction for all pause time values. For 30 sources, LBAR achieves packet delivery fractions between 85% and 100%. AODV, on the other hand, achieves packet delivery fractions between 79% and 91%, while DSR achieves from 72% to 93%. LBAR achieves 8% - 19% higher packet delivery fraction than AODV and 6% - 20% than DSR. This is mainly because of redundant route information that is stored in the destination node to provide aid in routing, which eliminates the necessity of source reinitiation of route discovery. When an upstream node on an active path cannot hear its downstream neighbor, this node notifies the destination to check the routing table to look for an alternative path through itself. When that node receives acknowledgement with a new path from the destination, it detours all associated packets along the new path towards the destination (See Figure 3.2). In contrast, AODV and DSR adopt the mechanism of dropping packets and informing the source node to restart route discovery, once the next hop becomes unreachable on an active path. Therefore, LBAR delivers more packets to their destinations than AODV and DSR. Similarly, LBAR has superior performance to both

AODV and DSR in the case of 40 sources, in terms of the packet delivery fraction (see Figure 4.1 (d)).

Also, LBAR has a better average end-to-end delay than both AODV and DSR with 10, 20, 30 and 40 sources (see Figure 4.2 (a), (b), (c) and (d)). For 10 and 20 sources, the difference of average end-to-end delay among these three protocols is not very noticeable although LBAR has better performance in terms of average delay. However, for 30 and 40 sources, LBAR achieves significantly lower delay than AODV and DSR. In fact, LBAR outperforms AODV by a factor of 1.7 for lower pause times and 5.1 for higher pause times. Likewise, LBAR outperforms DSR by a factor of 2.9 for lower pause times and 5.5 for higher pause times. Moreover, the delays decrease with lower mobility for LBAR in all four cases while it increases with 30 and 40 sources for both AODV and DSR (Figure 4.2 (c) and (d)). This is due to a high level of network congestion and multiple access interference in certain regions of the ad hoc network. Neither AODV nor DSR has any mechanism for load balancing, i.e., for choosing routes in such a way that the data traffic can be more evenly distributed in the network. This phenomenon is less visible with higher mobility where traffic automatically gets more evenly distributed due to source movements. In contrast, LBAR adopts a mechanism for load balancing, which tries to route packets along a less congested path to avoid overloading some nodes. This mechanism is based on the concept of nodal activity as defined in Section 3.1.4. The larger the activity of the node, the more the load of the node is. LBAR route discovery

protocol uses request messages to find a path with relatively less traffic, i.e., the path with least total nodal activity value. This explains the very low average delay with mobility variation.

In all cases (Figure 4.3 (a), (b), (c) and (d)), we notice the routing load of these three protocols increases with increasing the number of sources. This is because the increase in the number of source nodes causes a greater number of request messages flooding. LBAR demonstrates a higher routing load than both AODV and DSR. AODV and DSR only accept the first request message at every node, that is, if a node has already seen a request message for a particular packet, it will not accept a second message of the same packet. On the other hand, LBAR accepts request messages as long as they are not looping through the node. (When request messages are routed, all request messages are assumed to contain a route record, including a list of all nodes IDs used in establishing the path fragment from the source node to the current intermediate node.) Destination nodes keep a record of different route information from request messages as backup for use during the path maintenance protocol. Therefore, LBAR will almost always have an alternative path to detour packets in case of link failure. This, however, comes at the expense of more routing load. Also, note that relative to LBAR and AODV, DSR always has lower routing load. By virtue of aggressive DSR route caching [8], DSR rarely resorts to a route discovery process unlike LBAR and AODV. Although DSR provides a significant benefit up to a certain extent, stale routes are often chosen as the route. Picking stale routes

causes three problems – (i) consumption of additional network bandwidth and interface queue slots even though the packet is eventually dropped or delayed; (ii) a lot more packets would be dropped on the stale route, (iii) higher delays due to retransmission of lost packets.

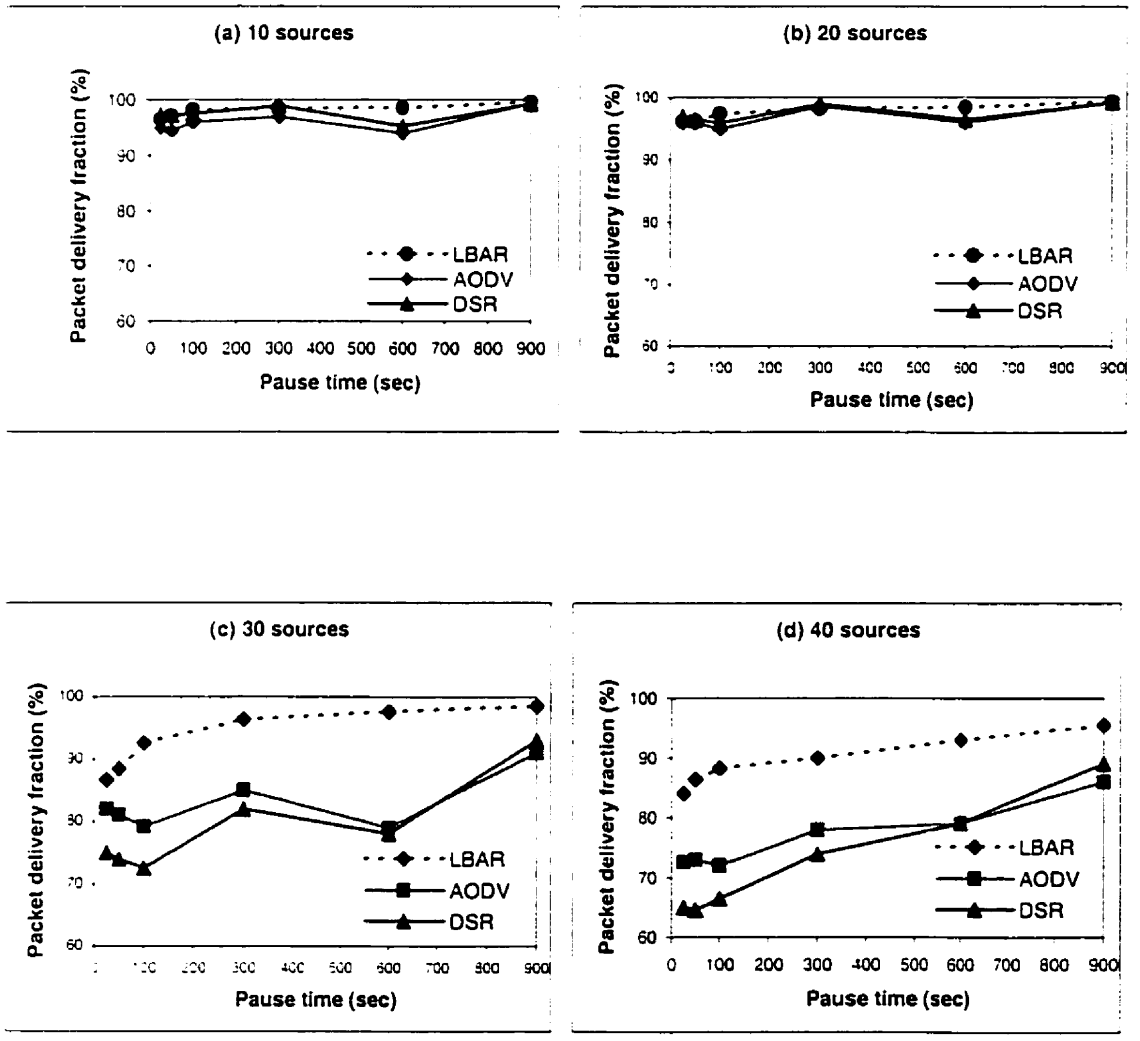

Figure 4.1: Packet delivery fraction for the 50-node model

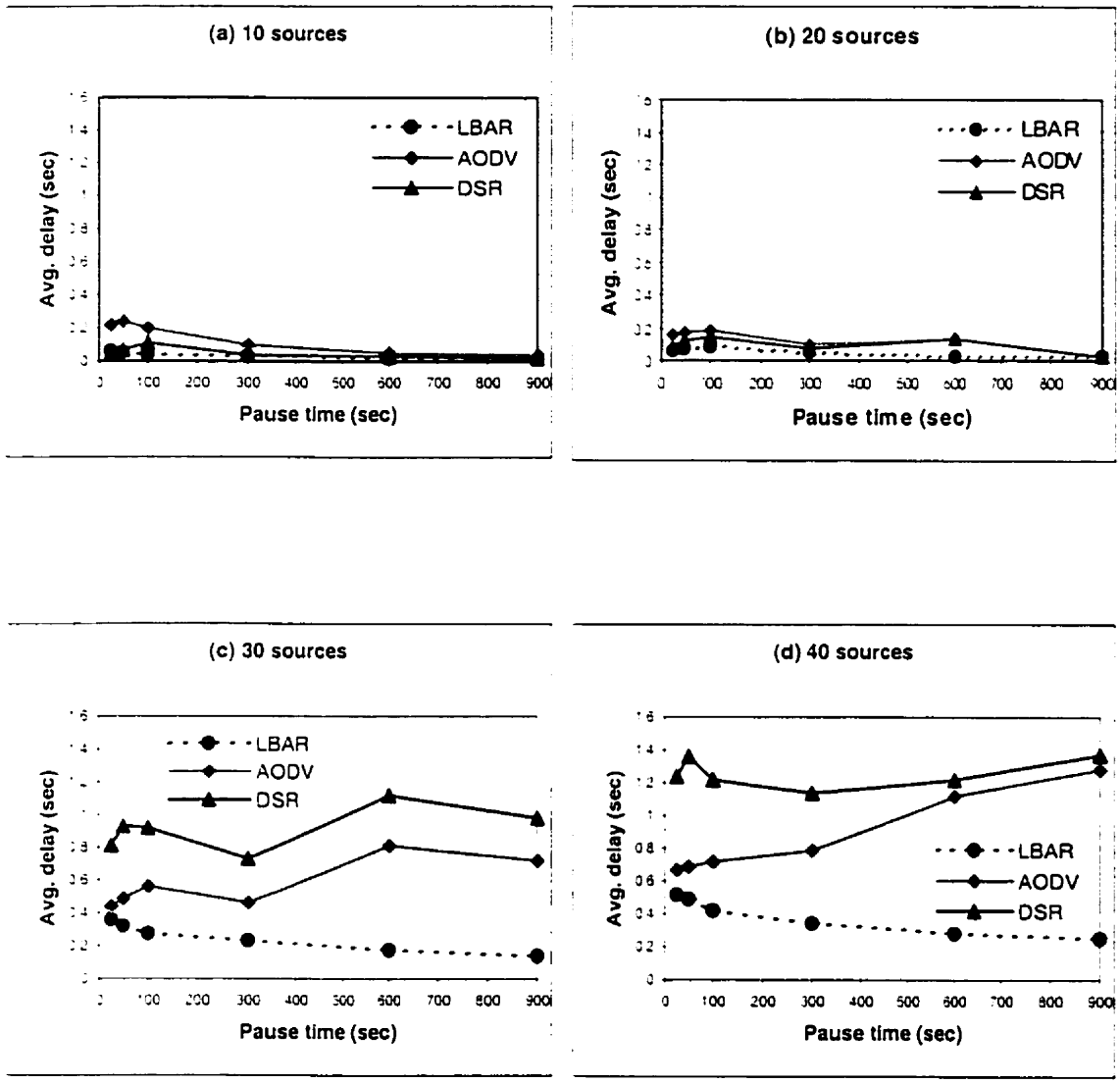

Figure 4.2: Average end-to-end delay for the 50-node model

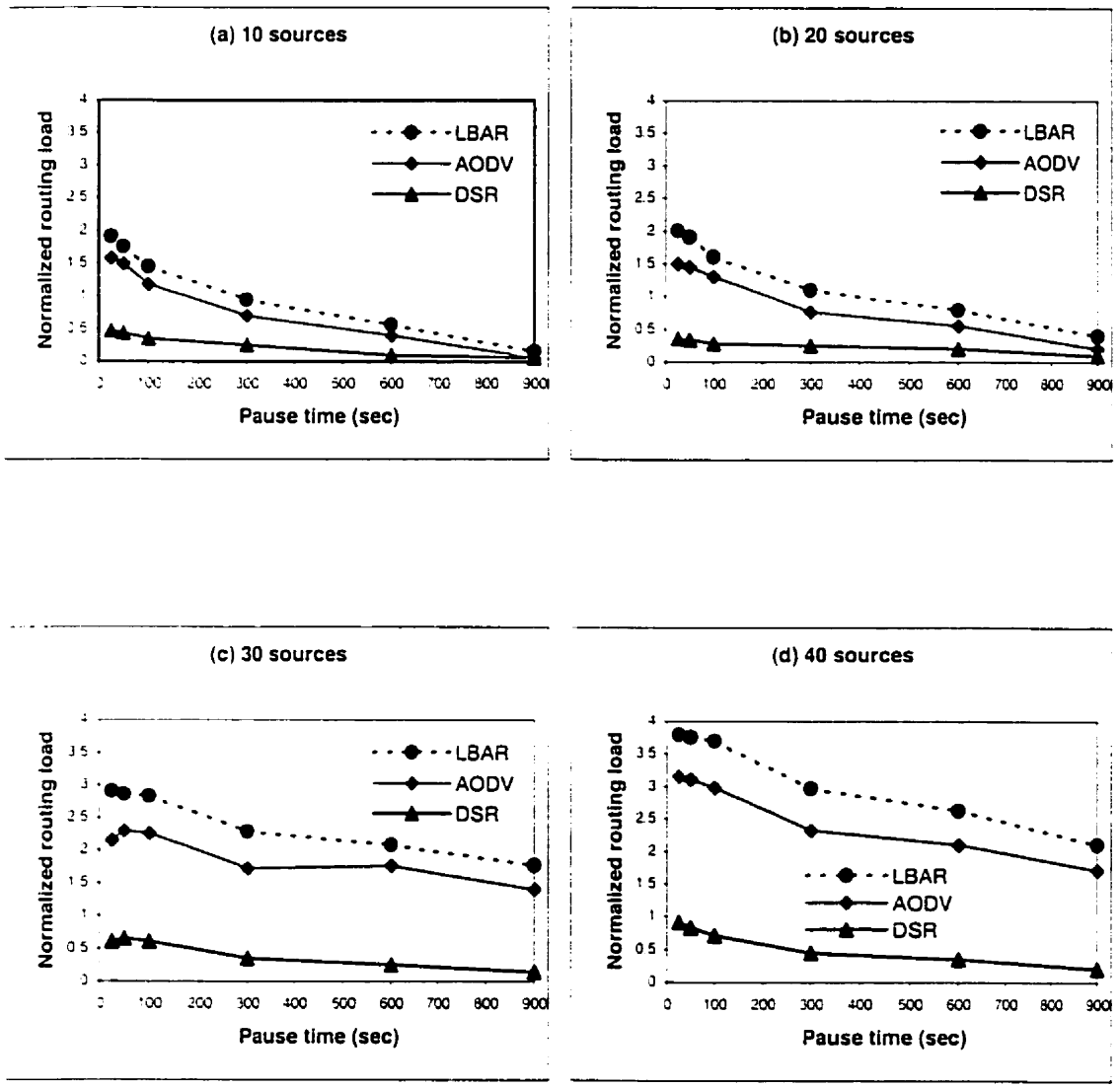

Figure 4.3: Normalized routing loading for the 50-node model

For the 100 node experiments, we used 10, 20 and 40 sources. The packet rate is fixed at 4 packets/sec for 10 and 20 sources, and 2 packets/sec for 40 sources. LBAR records higher packet delivery fractions than AODV and DSR (See Figure 4.4 (a), (b), (c)). Note that LBAR has similar packet delivery performance as AODV and DSR for 10 sources, achieving packet delivery fraction of over 90% (See Figure 4.4 (a)). However, the performance of both AODV and DSR degrades significantly in comparison with LBAR with larger number of sources (Figure 4.4 (b) and (c)). With 40 sources, LBAR achieves packet delivery fractions between 84% and 97%, compared to 74% and 96% for AODV and 48% and 95% for DSR. Moreover, packet delivery fraction of LBAR is only 4% less than that achieved for 10 sources at high mobility, as opposed to 14% for AODV and 30% for DSR. In particular, DSR loses about twice as many packets than LBAR for higher mobility scenarios. This supports the hypothesis that LBAR is more reliable, in terms of packet delivery, than AODV and DSR with varying node densities, and under variable network loads. The packet delivery performance of LBAR does not degrade when the number of the mobile nodes increases. This is largely due to the fact that LBAR redundant route mechanism plays an important role, as the destination would promptly respond to the node detecting link failure by an acknowledge message with a new route selected from its redundant routes. When the node that detects link failure receives the acknowledge message, it will transmit packets associated with the destination along that new path. In this way, packets would not be dropped as in AODV and DSR. This

redundant route mechanism ensures that packet routing will not be dramatically affected as in AODV and DSR when the number of mobile nodes increases.

Compared to the 50-node case, the performance of AODV and DSR degrades more than LBAR in terms of packet delivery fraction. We explain this as follows. When the number of mobile nodes increases, many possible valid routes to the destination will exist at the time of route discovery, and packets would go through many nodes to reach the destination. However, since the probability that the destination and intermediate nodes move to another location is high, by the time the route reply is received at the source and the source transmits its packets, it is possible that the node along the route might have moved. Therefore, the packet will not be delivered. On the contrary, LBAR performs quite well in both cases, whether under high-mobility or low-mobility conditions.

As shown in Figure 4.5, LBAR always has much lower average end-to-end delay than both AODV and DSR, which have degraded performance with larger number of sources. Also, we notice that the delay always decreases with lower mobility for LBAR, while varies with lower mobility for AODV and DSR. For 10 sources, the difference in delay results is not prominent. When the number of source nodes increases, the difference is more noticeable. For 20 sources, similar the 50-node case, LBAR has a lower delay than AODV and DSR by a factor of 2.4 and 6.0, respectively. This can be also accredited to the LBAR load-balancing mechanism which tries to route packets along a path with less

traffic to alleviate congestion. With this effort, packets will almost always be transmitted to the destination faster, and data traffic tends to be more evenly distributed in the network than AODV and DSR.

Note that LBAR achieves a more stable average delay than AODV and DSR both in the 50-node and the 100-node case. For example, consider the 20-source model in Figure 4.2 (b) and Figure 4.5 (b). We notice that the delay of LBAR only goes up by a factor of 2.2 while AODV grows by a factor of 2.7 and DSR soars up by a factor of 8.5. This demonstrates that the average delay of LBAR is much more stable with varying the size of the network than AODV and DSR.

The performance of the normalized routing load with different source nodes for the 100-node case is shown in Figure 4.6. Routing load of LBAR, AODV and DSR will suffer from an increase when the network density, i.e., the number of nodes in the network increases. Increments to route request broadcast (because the larger the number of mobile nodes in the network, and hence in any one neighborhood, the larger the number of forwarded route requests) will contribute to increasing the routing load. As in the 50-node case, LBAR still has higher routing load than AODV and DSR. LBAR, however, achieves higher packet delivery fractions and lower average end-to-end delays.

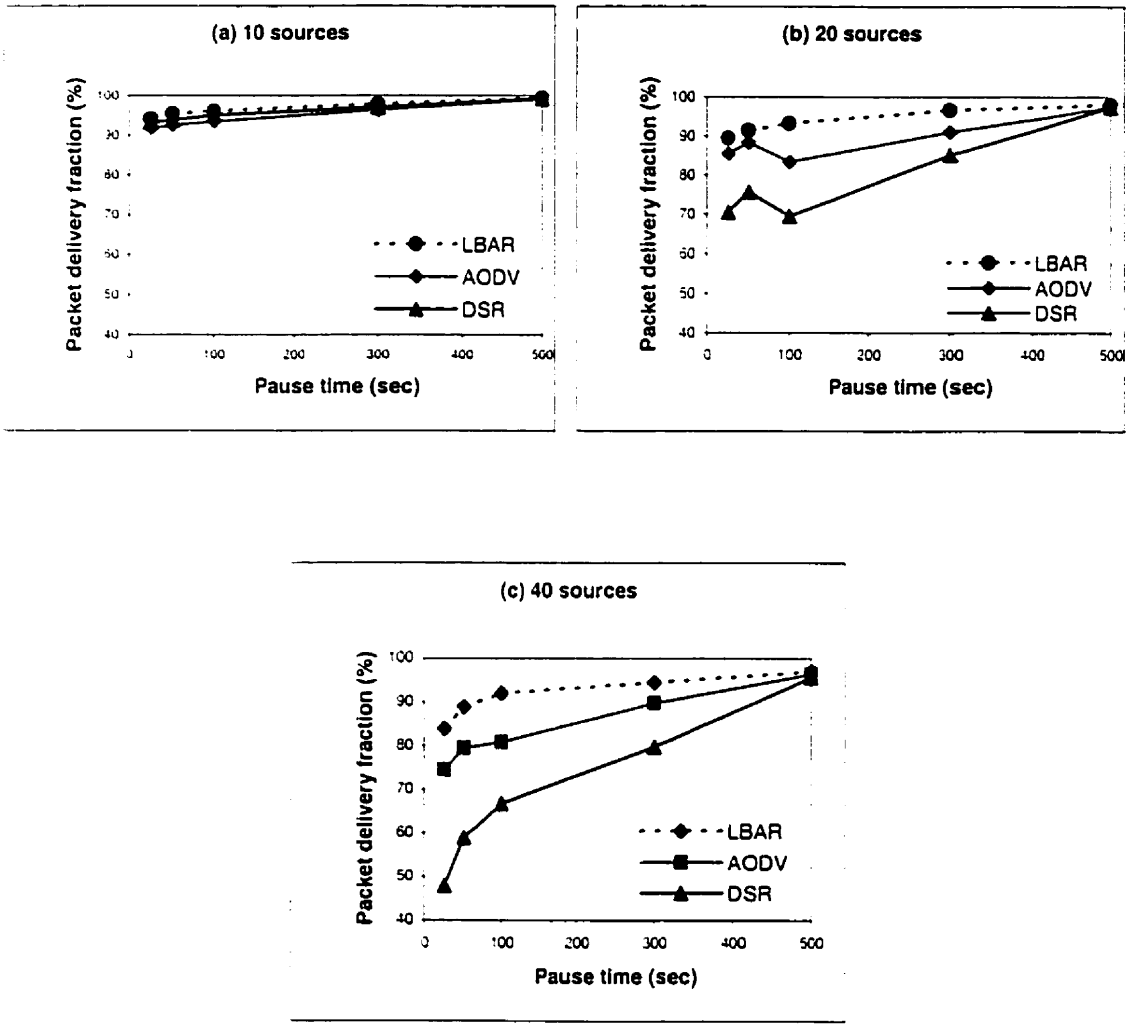

Figure 4.4: Packet delivery fraction for the 100-node model

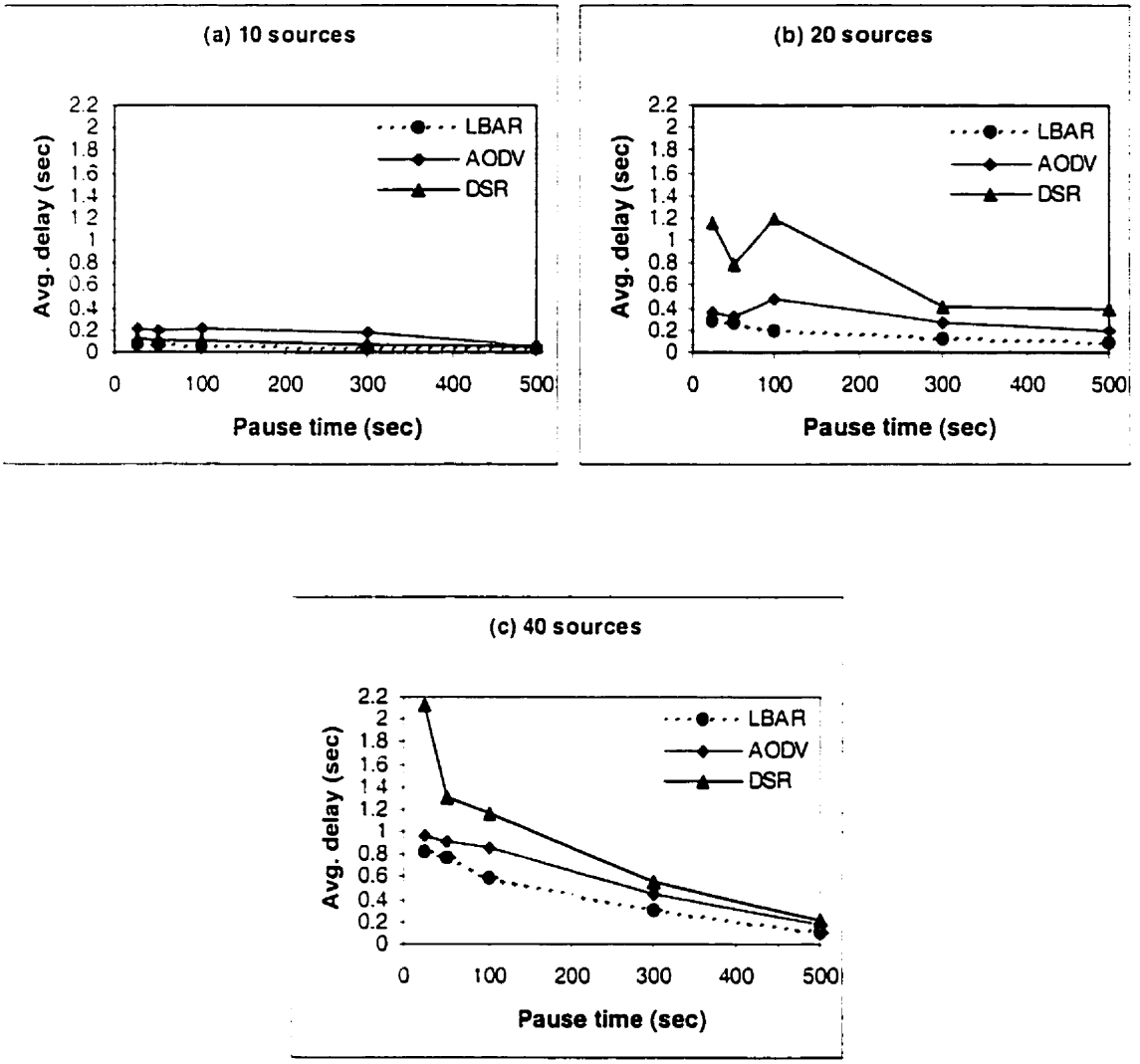

Figure 4.5: Average end-to-end delay for the 100-node model

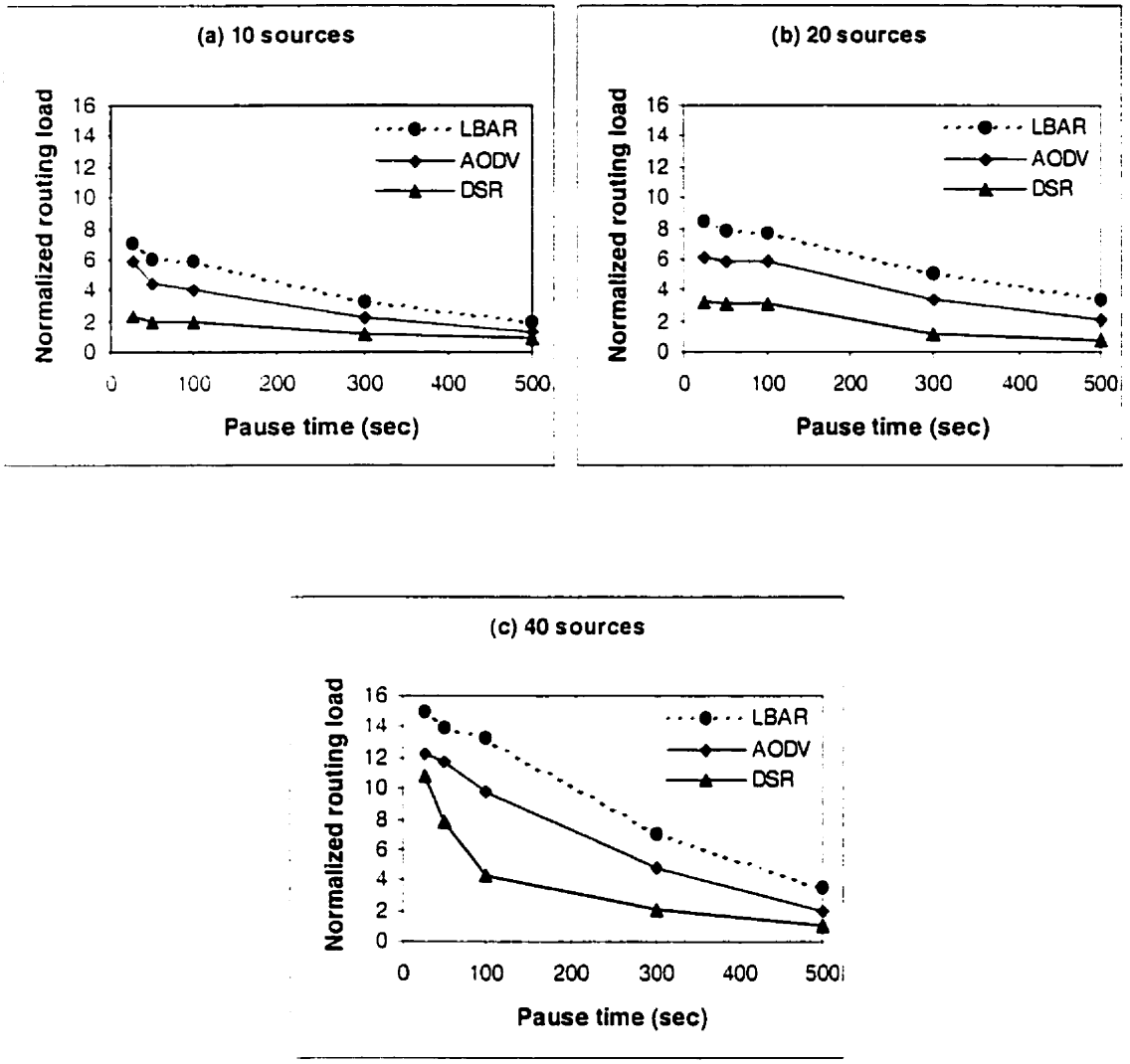

Figure 4.6: Normalized routing load for the 100-node model

4.4 Summary

This chapter evaluated the performance of our proposed LBAR protocol, through a comprehensive simulation model. The performance of LBAR is compared to that of AODV [9] and DSR [8]. The adopted traffic and mobility model with the experimental setting used in the simulation is described. Several simulation experiments were conducted to investigate the effects of traffic source, node mobility and network density (the number of nodes in the network) on the performance of LBAR compared with the other two schemes.

Simulation experiments show that LBAR outperforms AODV and DSR with respect to the packet delivery fraction. LBAR demonstrates superior packet delivery fractions over those achieved by AODV and DSR, under different mobility and node densities. For the 100-node model with 20 CBR-sources, for example, LBAR packet delivery fractions are up to 12% higher than AODV, and 34% higher than DSR. The difference is even more prominent in the case of 40 sources, where LBAR outperforms AODV by 14%, and DSR by 38% at high mobility. In addition, LBAR has significantly lower average end-to-end delay. Under different scenarios, LBAR always achieves lower average delay than AODV and DSR. Moreover, LBAR's delay decreases with lower mobility scenario. This is a desirable property for efficiency of the protocols and stability of the network. On the other hand, LBAR requires fairly higher routing load than AODV and DSR.

In summary, LBAR achieves better performance than AODV and DSR for the application-oriented metrics (packet delivery fraction and average end-to-end delay). Although LBAR has a higher routing overhead than AODV and DSR, from the application perspective, transmission reliability and packet delay are the most important concerns.

Chapter 5

Conclusion

Ad hoc networks are primarily different from wired and other wireless networks in terms of the lack of an infrastructure. The rapidly changing network topology drastically and unpredictably makes inevitable the need for efficient dynamic routing protocols that must be able to build reliable communication between mobile nodes and keep up with the high degree of node mobility. Additionally, the services carried over ad hoc networks are expected to require low end-to-end delays. To face such challenges, many routing protocols have been proposed for wireless ad hoc networks in recent years. We have studied the pros and cons of some of the most prominent proposed routing protocols. None of the protocols, however, achieves load balancing to even traffic load over the network and consequently relieves congestion.

In this thesis, we have proposed a novel on-demand routing scheme, namely the Load-Balanced Ad-hoc Routing (LBAR) protocol. Unlike table-driven routing protocols,

mobile nodes are not required to keep up-to-date information about the whole network. Indeed, LBAR only builds routes based on request packets from source nodes. In LBAR route discovery phase, routing information on different paths is forwarded through setup messages to the destination. Setup messages carry up-to-date route cost within the traversed path. The destination node collects the information received on different paths, and then selects the path with the minimum cost, which is measured by nodal activity. By weighing total nodal activity of a path, congested paths can be avoided, as packets are and be transmitted along the least-activity path. As a consequence, traffic over the ad hoc network tends to be evenly distributed in the long term. In addition, in order to keep up with frequent topology change, LBAR provides quick response to link failure by patching up the broken routes in use, thus guaranteeing reliability of data transmission.

A simulator has been developed to demonstrate and study the performance of the proposed LBAR protocol. The results are compared to those of the AODV [9] and DSR [7][8] protocols. Simulation results have clearly shown the advantages of LBAR over AODV and DSR in terms of packet delivery fraction and average end-to-end delay. Therefore, LBAR presents itself as a powerful candidate as the routing protocol in wireless networks.

The load-balancing ad hoc routing protocol proposed in this thesis is mainly intended for connectionless applications. Many applications, however, require connection-oriented

communication with end-to-end quality of service guarantees. Such applications include, streaming media, video on demand and others. QoS connection management for connection-oriented applications in wireless ad hoc networks requires the interaction of three functions, namely, routing, call admission control (CAC) and medium access control (MAC). Future work include the development of QoS-based connection management techniques for ad hoc networks that are capable of relaying path/link quality information to the source node so that the network can apply CAC and make a decision whether to accept or reject a connection. While in conventional networks, CAC and routing may be performed separately; this would result in large connection setup times in ad hoc networks. This is due to the mobility and high variability of available resources in such networks. As well, this may add to the congestion of limited-bandwidth wireless channels. It should be noted that effective QoS-based routing algorithms for wireless ad hoc networks require the development of QoS-aware MAC protocols that are capable of assigning bandwidth to traffic flows in the network. This is essential for both routing and connection management.

Bibliography

[1] C.E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers", *Computer and Communication*, Oct. 1994, pp.234-244.

[2] C. -C. Chiang, "Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel" Proc. *IEEE SINGAPORE CONFERENCE – IEEE SICON'97*, Apr.1997, pp.197-211.

[3] Tsu-Wei Chen and Mario Gerla, "Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks" *IEEE International Conference on Communications (ICC), 1998.*

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

[5] Chai-Keong Toh, "A novel distributed routing protocol to support Ad hoc mobile computing" Proc. *1996 IEEE 15th Annual International Phoenix Conference Computer and Communication*, Mar. 1996, pp. 480-86.

[6] C. -K. Toh, "Long-lived Ad-Hoc Routing based on the concept of Associativity" *Internet Engineering Task Force -- IETF Draft*, Mar. 1999.

[7] David B. Johnson, Davis A. Maltz, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks" *IETF Draft*, Oct. 1999.

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

[9] Charles E. Perkins, Elizabeth M. Royer, Samir R. Das, "Ad Hoc On-demand Distance Vector Routing". *IETF Draft*, Oct. 1999.

[10] V.H. VacDonald. "The celluar concept". *The bell Systems Technical Journal*, vol. 44, 1965, pp. 547 -588

[11] S. Ramanathan and Martha Steenstrup, "A Survey of routing Techniques for Mobile Communications Networks", *Mobile Networks and Applications*, 1996, vol. 1, no. 2, pp. 89-104.

[12] R. Dube et al., "Signal Stability based adaptive routing for Ad Hoc Mobile networks", *IEEE Personal Communications Magazine*, Feb. 1997, pp. 36-45.

[13] G.R.Ash, *Dynamic Routing in Telecommunications Networks*, McGraw Hill, 1998.

[14] R. Perlman, *Interconnections*, Second Edition: Bridges, Routers, Switches, and Internetworking Protocols, 1999.

[15] 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, Aug. 1999, pp.1369-79.

[16] Mingliang Jiang, Jinyang Li, Y.C. Tay, "Cluster Based Routing Protocol" *IETF Draft*, Aug. 1999.

[17] Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang, "MACAW: A media access protocol for wireless LAN's", *In Proceedings of the SIGCOMM*, Aug. 1994, pages 212 – 225.

[18] IEEE Standards Department. *Wireless LAN medium access control (MAC) and physical layer (PHY) specifications*. IEEE standard 802.11, 1997.

[19] Z. J. Haas, "A New Routing Protocol for the Reconfigurable Wireless Networks," *Proc. IEEE International Conference on Universal Personal Communications (IEEE ICUPC)*, San Diego, CA, Oct. 1997, pp. 562-66.

[20] J. Moy, "OSPF Version 2", *IETF RFC 1583*, 1994.

[21] S. R. Das, C. E. Perkins and E. M. Royer, "Performance comparison of two on-demand routing protocols for ad-hoc networks," *IEEE INFOCOM*, March 2000.

[22] IETF MANET Working Group Charter http://www.ietf.org/html.charters/manet-charter.html.

[23] Ahmed M. Safwat, " Infrastructure-based routing in wireless mobile ad-hoc networks " MSc. Thesis. Department of Computing and Information Science Department, Queen's University, Apr. 2000.

Appendix

Confidence Intervals

Simulation results are estimates. so we need to specify boundaries by which we can express our confidence in the results. Normally, confidence intervals placed on the mean values of simulation results are donated to express the precision of these results.

Consider the results of N independent simulation runs for the same experiment: X_1, X_2, X_3, ... , X_N. Additionally, we assume that these results are statistically independent. The sample mean, $\overline{X}_N$, of these results is given by

$$\overline{X}_N = \frac{1}{N} \sum_{i=1}^{N} X_i$$

and the variance of the distribution of the sample values, S_X^2, is donated by

$$S_X^2 = \frac{1}{N-1} \sum_{i=1}^{N} (X_i - \overline{X}_N)$$

The standard deviation of the sample mean is given by

$$\frac{S_X}{\sqrt{N}}$$

The mean of simulation runs may fall in the interval $\pm\varepsilon$ within the actual mean with a certain probability drawn from the t-distribution,

$$\varepsilon = \frac{S_X t_{\alpha/2,N-1}}{\sqrt{N}}$$

where $t_{\alpha/2,N-1}$ is the value of the t-distribution with N-1 degrees of freedom with probability $\alpha/2$.

The confidence intervals with respect to the simulation results have upper and lower limits, which are defined as follows:

Lower Limit = $\overline{X}_N - \varepsilon$

Upper Limit = $\overline{X}_N + \varepsilon$

In this thesis, a 90% confidence level was used. 10% confidence intervals for each data point were obtained. The simulation running time has been chosen long enough to ensure stability and tight confidence intervals.