PRO-ACTIVE ROUTE MAINTENANCE IN DSR

by

Liang Qin

A thesis submitted to the Faculty of Graduate Studies in partial fulfillment of the requirement for the degree of

Master of Science

Ottawa-Carleton Institute of Computer Science

School of Computer Science
Carleton University
Ottawa, Canada

August 2001

The undersigned recommend to the Faculty of Graduate Studies and Research
acceptance of the thesis

PRO-ACTIVE ROUTE MAINTENANCE IN DSR

Submitted by **Liang Qin**, M.E.
in partial fulfillment of the requirements for
the degree of M. Sc. in Information & System Science

Thesis Supervisor

DIRECTOR, SCHOOL OF COMPUTER SCIENCE

Carleton University

August 2001

ABSTRACT

Existing on-demand ad hoc network routing protocols continue using a route until a link breaks. During the route reconstruction, packets can be dropped, which will cause significant throughput degradation. In this thesis, we add a link breakage prediction algorithm to one on-demand routing protocol: the Dynamic Source Routing (DSR) protocol. The mobile node that implements the prediction algorithm uses signal power strength from the received packets to predict the link breakage time, and sends a warning to the source node of the packet if the link is soon-to-be-broken. The source node can perform a pro-active route rebuild to avoid disconnection. Experiments demonstrate that adding link breakage prediction to DSR, even considering the increased number of control messages (at most 33.5%), can significantly reduce the total number of dropped data packets (at least 20%) due to link breakage by reducing the number of broken links. We believe that TCP can potentially benefit well from the pro-active route maintenance to increase throughput, which is affected by broken links. We also propose a modification plan for AODV and make recommendations about further improvement on DSR based on the link breakage prediction.

ACKNOWLEDGEMENT

I would like to thank my supervisor, Professor Thomas Kunz, for his guidance and direction for this thesis, for the many interesting discussions we had. I greatly benefit from his detailed comments and insights that help me clarify my ideas and present the materials in a suitable way.

I would like to thank the teachers at the School of Computer Science and Department of Systems and Computer Engineering, Carleton University for all the things I learned here. I would also like to thank Mr. John Knox for his assistance and support of computer system in the lab, and my classmates in the lab for their helpful discussion.

The financial support of this project from Communication and Information Technology Ontario, and Natural Science and Engineering Research Council of Canada are gratefully acknowledged.

Finally I leave my special thanks to my wife Ying and my parents for their understanding and support.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ABR	Associativity Based Routing
AODV	Ad-hoc On-demand Distance Vector Routing
ARP	Address Resolution Protocol
CBR	Constant Bit Rate
CGSR	Cluster Gateway Switching Routing
DAG	Directed Acyclic Graph
DCM	Distributed Coordination Function
DSR	Dynamic Source Routing
FORP	Flow Oriented Routing Protocol
GPS	Global Position System
IMEP	Internet MANET Encapsulation Protocol
IP	Internet Protocol
LCC	Least Cluster Change
LL	Link Layer
LMR	Lightweight Mobile Routing
MAC	Media Access Control
MANET	Mobile Ad Hoc Network
NPDU	Network Protocol Data Units
NS2	Network Simulator
SSA	Signal Stability-based Adaptive Routing
TCP	Transmission Control Protocol
TORA	Temporally Ordered Routing Algorithm

Chapter 1 Introduction

In recent years, mobile computing has enjoyed a tremendous rise in popularity. The continued miniaturization of mobile computing devices and the extraordinary rise of processing power available in mobile laptop computers combine to put more and better computer-based applications into the hands of a growing segment of the population.

A "mobile ad hoc network" (MANET) is an autonomous system of mobile routers (and associated hosts) connected by wireless links - the union of which forms an arbitrary graph. The routers are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Such a network may operate in a standalone fashion, or may be connected to the larger Internet [1]. In ad hoc networks, mobile computer users with (compatible) wireless communication devices are allowed to set up a possibly short-lived network just for the communication needs of the moment. Example applications of ad hoc network applications include students using laptop computers to participate in an interactive lecture, business associates sharing information during a meeting, home networking, embedded computing applications, soldiers relaying information on the battlefield, and emergency services such as disaster relief personnel coordinating efforts after an earthquake.

Since IP is designed for stationary hosts in the Internet, Mobile IP [2] only considers the single-hop wireless case where the mobility patterns are relatively simple. In ad hoc networks, each node also acts as a router, forwarding data packets for other nodes. A central challenge in the design of ad hoc networks is the development of dynamic routing protocols that can efficiently find routes between two communicating nodes. The routing

protocol must be able to keep up with the realistically high degree of node mobility that often changes the network topology drastically and unpredictably.

Existing ad hoc routing basically can be divided into two categories: proactive and reactive, based on whether nodes should keep track of routes to all possible destinations, or instead keep track of only those destinations of immediate interest. Proactive protocols, which are also called table driven, keep track of all destinations in the ad hoc network. It has the advantage that communication with arbitrary destinations experience minimal initial delay from the point view of the application, but suffers the disadvantage of additional control traffic to continually update stale route entries, even when no applications are using these routes. This wasted effort can cause scarce bandwidth resources to be wasted and can cause further congestion. Destination-Sequenced Distance Vector (DSDV) [3] is an example of a proactive protocol. The reactive, or on-demand, protocols have been designed so that routing information is requested only when it is actually needed. It may use far less bandwidth for maintaining the route tables at each node, but the latency for many applications will increase because a route will have to be acquired before communication can begin. Ad Hoc On-demand Distance Vector (AODV) [4] and Dynamic Source Routing (DSR) [5] are examples of reactive protocols.

1.1 Motivation

In ad hoc networks, wireless media are of limited and variable range, in distinction to existing wired media. Each host moves in an arbitrary manner and routes are subject to frequent disconnection. During the period of route reconstruction, packets can be

dropped. The loss of packets will cause significant throughput degradation for both real-time and non real-time data. This effect on TCP is described in [6]. At the time when this thesis began, few algorithms and protocols tried to improve performance by using link state information. Associativity based routing (ABR) proposed by Toh [7] favors routes with longer-lived links according to the associativity of the incident nodes. He proposed a link state prediction model based on the knowledge of mobile nodes' position [8]. Flow Oriented Routing Protocol (FORP) [9] is the only protocol published then that also uses mobility host's position to predict link state. It handoffs when a link is predicted to be broken. But FORP did not demonstrate its advantage relative to two prominent on-demand routing protocols: AODV and DSR. Also, FORP requires every mobile host to know its own position by using Global Position System (GPS), which is not a standard component in today's portable computers.

1.2 Research Overview and Contributions

This thesis concentrates on the link state prediction in ad hoc networks to reduce the likelihood of data packets being dropped. In most existing protocols, a mobile host will continue using the route until a link is broken. Instead, the prediction algorithm proposed in this thesis will use the received signal power measurements to predict the topological change in order to perform a route rebuild prior to the link breakage, therefore avoiding the data packets being dropped.

This thesis will not propose a new routing protocol; instead, it will suggest modifications to one of two popular on-demand routing protocols, AODV and DSR, by using the prediction algorithm. The simulation results reported in [10] [11] [12] show

that both protocols have comparable performance with high packet delivery ratio and low control message overhead. Implementing the prediction algorithm, the number of data packets dropped because of link breakage will be reduced, and the overall performance will be improved further.

This thesis provides the following contributions:

1. Propose a prediction algorithm which uses receive signal power strength to predict the link breakage. Simulations have been done to analyze the accuracy and limitation of the algorithm.
2. The prediction algorithm is implemented in DSR within a network simulator (NS2) to enable proactive route maintenance. (Please see Appendix for more information).
3. A comprehensive evaluation of the DSR protocol with prediction algorithm in terms of a specific set of performance metrics. The result shows that proactive route maintenance can significantly reduce total number of dropped packets due to link breakage (at least 20%), even considering the increased number of control messages (less than 33.5%).
4. Modifications for AODV are also being proposed. Suggestions are made for further improvement on avoiding packet drops and reducing the number of control messages.

1.3 Organization of Thesis

This thesis focuses on the link state prediction, its implementation and evaluation in the DSR protocol. Chapter 2 is a survey about existing wireless ad hoc networks routing protocols. Two categories of protocols are introduced; some simulation results are presented. Chapter 3 briefly introduces the network simulator (NS2), presents the

simulation results for two on-demand protocols, AODV and DSR, with different number of senders, mobility rate, and data rate, and analyzes the simulation results in terms of a specific set of performance metrics. Chapter 4 proposes the link state prediction algorithm based on the received signal power strength, algorithm parameter settings; analyzes the simulation results for the prediction accuracy and limitation. Chapter 5 presents the prediction algorithm implementation in DSR. Scenarios of exception are analyzed. We compare the simulation results under different mobility rates of DSR with and without prediction algorithm. Chapter 6 draws our conclusion; proposes a plan for the prediction algorithm implementation in AODV and suggests directions for future research.

Chapter 2 Background

2.1 Introduction

Mobile hosts and wireless networking hardware are becoming widely available. There are currently two variations of mobile wireless networks. The first is known as infrastructure networks, i.e. those networks with fixed and wired gateways. The bridges for these networks are known as base stations. A mobile unit within these networks connects to, and communicates with, the nearest base station that is within its communication radius.

The second type of mobile wireless networks is the infrastructureless mobile network, commonly known as an ad hoc network. A mobile ad hoc network is a network in which a group of mobile computing devices communicate among themselves using wireless radio, without the aid of a fixed networking infrastructure.

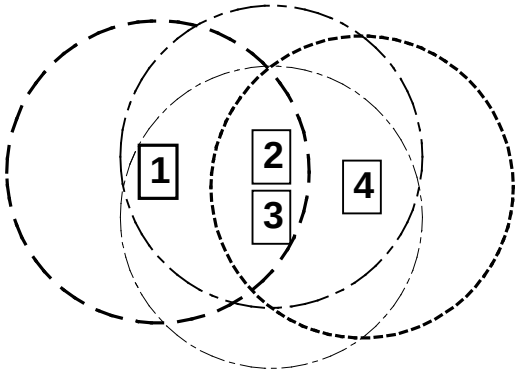


Figure 2.1 Ad Hoc Network with Four Wireless Nodes

Figure 2.1 illustrates an ad hoc network with four wireless mobile hosts. The circles represent the radio transmission range of the nodes. So node 1 and node 4 cannot send packets to each other directly. There are multiple paths between nodes 1 and 4; packets may travel the path 1-2-4, 1-3-4, or even 1-3-2-4.

In the ad hoc wireless network, the transmission range of each node is limited by power constraints, frequency reuse and channel effects. Consequently, store-and-forward packet routing is required over multiple-hop wireless paths.

A MANET consists of mobile platforms (e.g., a router with multiple hosts and wireless devices), herein simply referred to as "nodes", and it has several salient characteristics:

- Dynamic topologies: Nodes are free to move arbitrarily; thus the network topology may change randomly and rapidly at unpredictable times. New nodes may unexpectedly join the network or existing nodes may leave or be turned off.
- Bandwidth-constrained, variable capacity links: wireless links will have significantly lower capacity than their wired counterparts. In addition, the realized throughput of wireless communications is often much less than a radio's maximum transmission rates.
- Energy-constrained operation: The nodes in a MANET may rely on battery. For these nodes, energy conservation must be considered in system design.

Routing protocols in conventional wired networks generally use either distance vector or link state algorithms, both require periodic messages to be broadcast by each router. In distance vector routing, each router broadcasts to each of its neighbor routers its view of the distance to all hosts, and each router computes the shortest path to each host based on the information advertised by each of its neighbors. In link state routing, each router instead broadcasts to all other routers in the network its view of the status of each of its adjacent network links. Each router then computes the shortest distance to each host based on the complete picture of the network formed from the most recent link

information from all routers.

Traditional routing protocols have not been designed specifically to provide the kind of dynamic, self-starting behavior needed for ad hoc networks. These protocols pass detailed topology information between hosts and are not effective in an ad hoc network due to the high rate of topology change. In a dynamic network, the topology information (in the routing tables) is soon out of date, and the propagation of the routes is too slow to be accurate. Moreover, the wireless medium differs in important ways from wired medium; for example, wireless media are of limited and variable range, in distinction to existing wired media. In ad hoc networks, each host moves in an arbitrary manner and routes are subject to frequent disconnection.

2.2 Existing Routing Protocols for Ad Hoc Wireless Networks

Multihopping, mobility, large network size combined with device heterogeneity, bandwidth and battery power limitations make the design of adequate routing protocols a major challenge. In recent years, various different routing protocols have been proposed for wireless ad hoc networks. As shown in Figure 2.2 [12], ad hoc routing protocols may be generally categorized as 1) table-driven and 2) source-initiated on-demand driven.

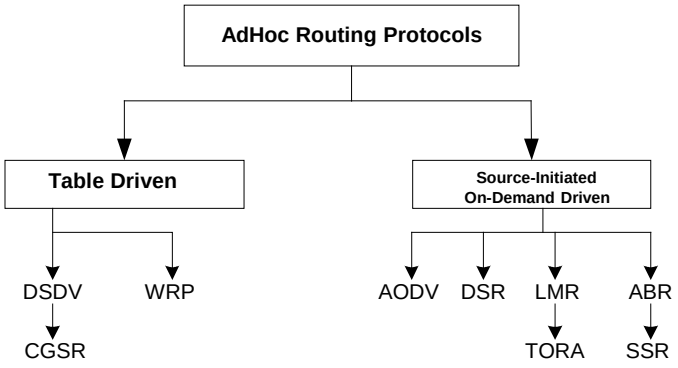


Figure 2.2 Categorization of Ad Hoc Routing Protocols

2.2.1 Table-Driven Routing Protocols

The 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.

2.2.1.1 Destination-Sequenced Distance-Vector Routing (DSDV)

The Destination-Sequenced Distance-Vector Routing (DSDV)[3] is based on the classical Bellman-Ford routing algorithm [18]. The key advantage of DSDV over traditional distance vector protocols is that it guarantees loop-freedom by tagging each route table entry with a sequence number to order the routing information.

In DSDV, each node maintains a routing table listing all available destinations. The attributes of each destination are the next hop, the number of hops to reach to the destination, and a sequence number, which is originated by the destination node.

DSDV uses both periodic and triggered routing updates to maintain table consistency. Triggered routing updates are used when network topology changes are detected in order to propagate the routing information as quickly as possible. To alleviate the potentially large amount of network traffic that such updates can generate, two types of packets are defined. The first is called a "full dump". This type of packets carries all available routing information and may require multiple network protocol data units (NPDU); the other type of packets will carry only information changed since the last full dump and should fit in

one NPDU in order to decrease the amount of traffic generated.

Mobile nodes cause broken links when they move from place to place. When a link to the next hop is broken, any route through that next hop is immediately assigned an infinity metric and an updated sequence number. This is the only situation when any mobile node other than the destination node assigns the sequence number. Sequence numbers assigned by the origination nodes are even numbers, and sequence number assigned to indicate infinity metrics are odd numbers. When a node receives an infinity metric, and it has an equal or later sequence number with a finite metric, it triggers a route update broadcast, and the route with infinity metric will be quickly replaced by the new route.

When a mobile node receives a new route update packet, it compares it to the information already available. Route updates are selected according to the following criteria:

- Routes are always preferred if the sequence numbers are greater;
- Otherwise, routes are preferred if the sequence numbers are the same and the metric is better.

The metrics for newly received routes are each incremented by one hop since incoming packets will require one more hop to reach the destination.

In an environment where many independent nodes transmit routing tables asynchronously, some fluctuations could develop. DSDV also uses settling time to prevent fluctuations of routing table updates. The settling time is used to decide how long to wait before advertising new routes.

The DSDV protocol guarantees loop-free paths to each destination and detects routes

very close to optimal. It requires nodes to periodically transmit routing update packets. These update packets are broadcast throughout the network. When the number of nodes in the network grows, the size of the routing tables and the bandwidth required to update them also grows, which could cause excessive communication overhead. This overhead is nearly constant with respect to mobility rate. As Broch et al. [10] found in their simulations, DSDV delivers virtually all data packets when node mobility rate and speed are low, and fails to converge as node mobility increases.

2.2.1.2 Cluster-Gateway Switching Routing (CGSR)

Cluster-Gateway Switching Routing (CGSR) [14] is based on a cluster based network architecture. In an ad hoc network, the mobile nodes are aggregated into clusters; each cluster is controlled by a cluster head. All nodes within transmission range of the cluster head belong to this cluster; so all nodes in a cluster can communicate with the cluster head, and (possibly) with each other. At the MAC layer, cluster heads are given a priority, and then they have more chances to transmit than other nodes, because they are in charge of broadcasting within the cluster and of forwarding packets between mobile nodes that are not “connected”.

The cluster head can be elected by using a distributed algorithm within a 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 election rather than packet relaying. The Least Cluster Change (LCC) algorithm is introduced to prevent frequent cluster head changes. According to the algorithm, only

two conditions cause the cluster head to change. One is when a node becomes disconnected from any cluster, the other is when two cluster heads come within range of each other.

Each node keeps two tables to route packets. One is the cluster member table which is used to map a destination address to the destination cluster head address, and the other is the routing table, which is used to select the next node to reach the destination cluster.

A gateway is a node belonging to more than one cluster. CGSR uses DSDV as the underlying routing scheme, and modifies DSDV by using a hierarchical cluster head-to-gateway routing to route packets from source to destination. So packets are routed alternatively through cluster heads and gateways, the typical route looks like $C_1G_1C_2G_2\ldots C_iG_i$, where C_i is a cluster head and G_i is a gateway. Because cluster heads have more chances to transmit than other nodes, the presence of the cluster head between two gateways is well worth the cost of the extra hop.

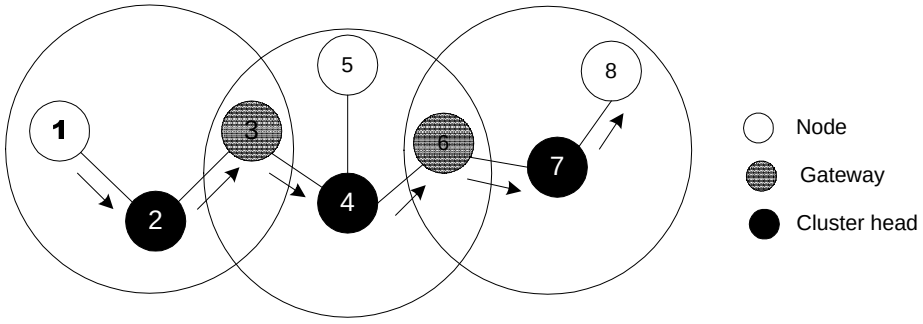


Figure 2.3 CGSR: Routing from Node 1 to Node 8

When a node receives a packet, it will select the shortest destination cluster head according to the cluster member table and routing table, and then it will select the next node to transmit for that destination cluster head according to the routing table. Figure 2.3

shows an example: node 1 sends a packet to node 8.

To maintain the route, each mobile node broadcasts its cluster member table periodically, and it updates its cluster member table when it receives a new one from its neighbor. The destination sequence number is used as in DSDV to avoid stale routes. Cluster maintenance protocols run continuously in the background in order to adjust to node movements and dynamically reconfigure the cluster structure accordingly.

The major advantage of CGSR is that only the routes to the cluster heads are maintained due to hierarchical routing. However, there is overhead associated with maintaining clusters and broadcasting cluster tables periodically by each node.

2.2.2 Source-Initiated On-Demand Routing Protocols

On-demand routing is the most recent entry in the class of scalable wireless routing schemes. It is based on a query-reply approach. This type of routing protocol creates routes only when desired by source nodes. When a node requires a route to a destination, it initiates a route discovery process within a 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 some form of a route maintenance procedure until either the destination becomes inaccessible along every path from the source or until the route is no longer desired.

2.2.2.1 Temporally-Ordered Routing Algorithm (TORA)

Temporally-Ordered Routing Algorithm (TORA) [15] [16] is a distributed protocol based on a " link reversal" algorithm. It guarantees that all routes are loop-free, and typically provides multiple routes to any source/destination pair that requires a route to alleviate congestion. The protocol is "source initiated" and creates a set of routes to a given destination only when desired. Routing optimality (shortest path routing) is considered of secondary importance. The protocol is designed to minimize reaction to topological changes. A key design concept of TROA is that it decouples the generation of potentially far-reaching control message propagation from the rate of topological changes. Such messages are typically localized to a very small set of nodes near the occurrence of the change. To accomplish this, nodes need to maintain routing information about adjacent nodes (one hop knowledge).

Since multiple routes are typically established, many topological changes require no reaction at all, because having one route is sufficient. Following topological changes that do require reaction, the protocol quickly re-establishes valid routes via a temporally-ordered sequence of diffusing computation: each computation consisting of a sequence of directed link reversals. Finally, in the event of network partition, the protocol detects the partition and erases all invalid routes within a finite time.

The protocol can be separated into three basic functions: creating routes, maintaining routes, and erasing routes. Three distinct control packets are used to accomplish these three functions: query (QRY) for creating routes, update (UPD) for both creating and maintaining routes, and clear (CLR) for erasing routes.

During the route creation and maintenance phase, nodes use a "height" metric to

establish a directed acyclic graph (DAG) rooted at the destination. The "height" associated with each node is a ordered quintuple $H_i = (\tau_i, oid_i, r_i, \delta_i, i)$, where τ_i is a time tag set to the "time" of the link failure; oid_i is the unique ID of the node which defined the new reference level; r_i is a reflection indicator bit, δ_i is a propagation ordering parameter; and i is the unique ID of the node itself. The first three elements in the quintuple represent the reference level and last two values represent a delta. The "height" is lexicographically ordered, so links are assigned a direction (upstream or downstream) based on the relative height metric of the neighboring nodes. During routing, a node may only route information to a lower node.

At each node in the network, a logically separate copy of the TORA algorithm is run for each destination. When a node needs a route to a particular destination, it broadcasts a QRY packet containing the address of the destination for which it requires a route. This packet propagates through the network until it reaches either the destination, or an intermediate node having a route to the destination. The recipient of the QRY then broadcasts an UPD packet listing its height with respect to the destination (if the recipient is the destination, the height is 0). As this packet propagates through the network, each node that receives the UPD sets its height to a value greater than the height of the neighbor from which the UPD packet was received. This has the effect of creating a series of directed links from the original sender of the QRY to the node that initially generated the UPD packet.

An underlying protocol IMEP, the Internet MANET Encapsulation Protocol [17], is designed to provide reliable, in-order delivery of all routing control messages from a node to each of its neighbors, plus notification to the routing protocol whenever a link to

one of its neighbors is created or broken. For link status sensing and maintaining a list of a node's neighbors, each IMEP node periodically transmit a BEACON packet, which is answered by each node hearing it with a HELLO packet.

When a node lost its last downstream link (i.e. becomes a local minimum), as a result of link failure, the node selects a new height such that it becomes a global maximum by defining a new "reference level", which is higher than any previously defined reference levels. Links are reversed to reflect the change for adapting to the new reference level. Figure 2.4 summarizes five cases starting from the time a node loses its last downstream link [16].

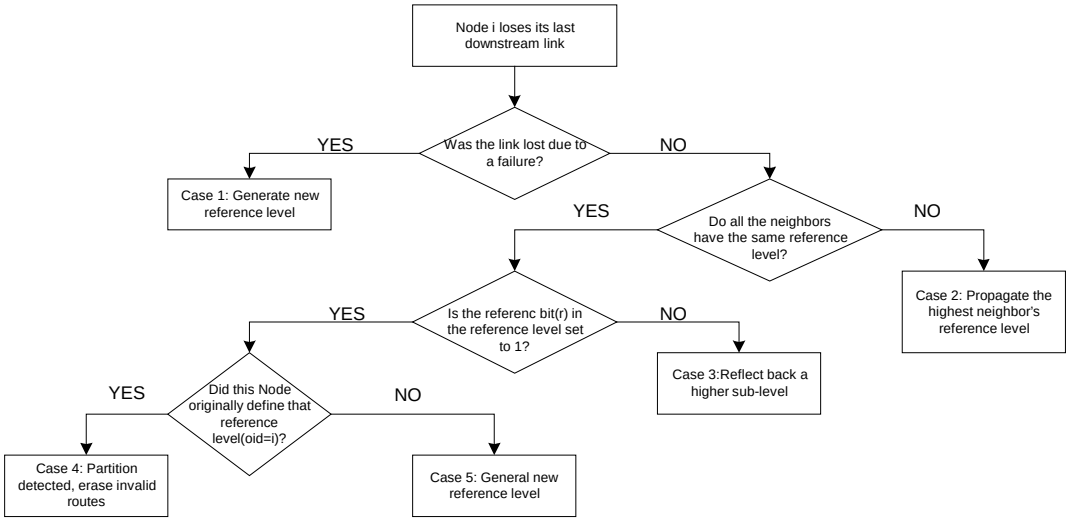


Figure 2.4 Maintaining Routes Decision Tree

Figure 2.5 is an example of re-establishing a route after failure of the last downstream link [16]. When a node detects a network partition (case 4), where a part of the network is physically separated from the destination, the node generates a CLR packet that resets the routing state and removes invalid routes from the network.

In TORA, in terms of memory requirement, each node must maintain a structure

describing the node's height as well as the status of all connected links per connection supported by the network. In terms of bandwidth requirement, each node must be in constant coordination with neighboring nodes in order to detect topology changes and coverage.

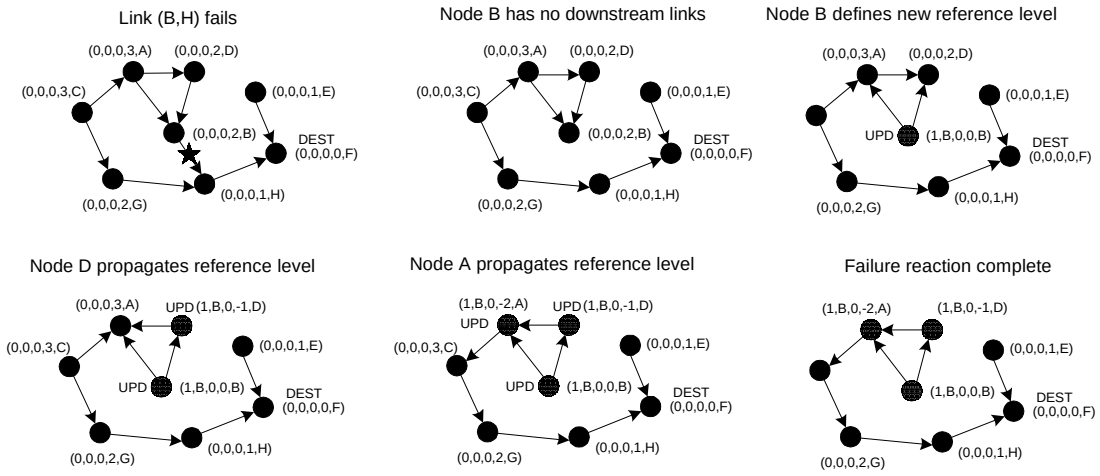


Figure 2.5 Re-establishing Routes after Failure of the Last Downstream Link

2.2.2.2 Dynamic Source Routing (DSR)

Dynamic Source Routing (DSR) [5] is based on the concept of source routing. There are no periodic routing advertisements in the protocol. Instead, when a node needs a route to another node, it dynamically determines one based on cached information or on the result of a route discovery process. Source routing is a routing technique in which the sender of the packet determines the complete sequence of the nodes through which to forward the packet. The sender explicitly lists this route in the packet's header, identifying each forwarding "hop" by the address of the next node to which to transmit the packet on its way to the destination host. A key advantage of source routing is that intermediate hops

do not need to maintain routing information in order to route the packet they receive, since the packets themselves already contain all the necessary routing information. Unlike conventional routing protocols, the DSR protocol uses no periodic routing advertisement messages, thereby reducing network bandwidth overhead, particularly during periods when little or no significant host movement is taking place.

The DSR protocol consists of two mechanisms: Route Discovery and Route Maintenance. When a mobile node wants to send a packet to some destination, it first checks its route cache to determine whether it already has a route to the destination. If it has one, which is not expired, it will use this route to send the packet. Otherwise, if the node does not have such a route, it will initiate route discovery by broadcasting a route request packet. This route request packet contains the addresses of the source node and the destination, and a unique sequence number “request id”, which is set by the source node. Each node in the network maintains a list of (source address, request id) pair that it has recently received from any host in order to detect duplicate route requests received. When receiving a request packet, if a node has already received this (source address, request id) pair or it finds its own address already recorded in the request, it discards the copy and does not process it further. Otherwise, it appends its own address to the route record in the route request packet and re-broadcasts the query to its neighbors. When the request packet reaches the destination, the destination node then sends a route reply packet to the source with a copy of the route. If a node can complete the query from its route cache, it may send a reply packet to the source without propagating the query packet further. Furthermore, any node participating in route discovery can learn routes from passing data packets and gather this routing information into its route cache. Figure

2.6 is an example of the creation of a route record in DSR [12].

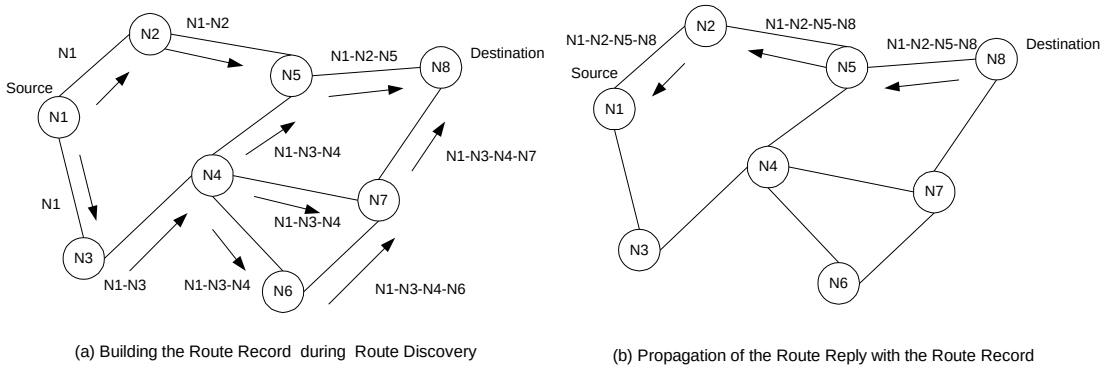


Figure 2.6 Creation of the Route Record in DSR

When sending or forwarding a packet to a destination, Route Maintenance is used to detect if the network topology has changed such that the route used by this packet is broken. Each node along the route, when transmitting the packet to the next hop, is responsible for detecting if its link to the next hop has broken. Many wireless MAC protocols, such as IEEE 802.11, retransmit each packet until a link-layer acknowledgement is received, or until a maximum number of retransmission attempts have been made. Alternatively, DSR may make use of a passive acknowledgement [19]. When the retransmission and acknowledgement mechanism detects that the link is broken, the detecting node returns a Route Error packet to the source of the packet. When a Route Error packet is received or overheard, the hop in error is removed from any route caches and all routes, which contain this hop, must be truncated at that point. The source can then attempt to use any other route to the destination that is already in its route cache, or can invoke Route Discovery again to find a new route.

There are several optimization options for the DSR protocol to reduce the latency and control message overhead [19]:

- Nonpropagating Route Requests: When performing Route Request, nodes first send a Route Request with the maximum propagation limit (hop limit) set to zero, prohibiting their neighbors from rebroadcasting it.
- Gratuitous Route Replies: A node may overhear a packet not addressed to itself, but its address is in the unprocessed portion of this packet. If so, the node knows that that packet could bypass the unprocessed hop preceding it in the source route. The node then sends a gratuitous Route Reply message to the packet's source, giving it the shorter route without these hops.
- Salvaging: When an intermediate node forwarding a packet finds that the next hop of the packet is broken, it checks its route cache for another route to the same destination. If a route exists, the node replaces the broken source route on the packet's header with the route from its cache and retransmits the packet, and returns a Route Error to the source of the data packet.
- Gratuitous Route Errors: When a source node receives a Route Error message, it will piggyback this bad link on its next Route Request message. In this way, stale information in the route caches around this source node will not generate Route Replies that contain the same bad link.

The DSR protocol is intended for networks in which the mobile nodes move at a moderate speed with respect to packet transmission latency [5]. An advantage of DSR over some on-demand protocols is that DSR does not use periodic routing advertisements, thereby saving bandwidth and reducing power consumption. On the other hand, as the network becomes larger, control packets and data packets also become larger because they need to carry addresses for every node in the path. Also, aggressive use of

route cache and the absence of any mechanism to expire stale routes will cause poor delay and throughput performance in more stressful situations [16].

2.2.2.3 Ad-hoc On-Demand Distance Vector Routing (AODV)

Ad-hoc On-Demand Distance Vector Routing (AODV) [4] is essentially a combination of both DSR and DSDV. It borrows the conception of sequence numbers from DSDV, plus the use of the on-demand mechanism of route discovery and route maintenance from DSR. It is called a “pure on-demand route acquisition system”; nodes that do not lie on active paths neither maintain any routing information nor participate in any periodic routing table exchanges. It is loop-free, self-starting, and scales to large numbers of mobile nodes.

When a source node needs to send a packet to a destination node for which it has no routing information in its table, the Path Discovery process is initiated. The source node broadcasts a route request (RREQ) to its neighbors. Each node that forwards the RREQ packet creates a reverse route for itself back to source node. Every node maintains two separate counters: a node sequence number and a broadcast_id. Broadcast_id is incremented when the source issues a new RREQ. Together with the source's address, it uniquely identifies a RREQ. In addition to the source node's IP address, current sequence number, and broadcast ID, the RREQ also contains the most recent sequence number for the destination of which the source node is aware. A node receiving the RREQ may send a route reply (RREP) if it is either the destination or if it has a route to the destination with corresponding sequence number greater than or equal to that contained in the RREQ. If this is the case, it unicasts a RREP back to the source. Otherwise, it

rebroadcasts the RREQ. Each node that participates in forwarding a RREP packet back to the source of RREQ creates a forward route to the source node. Each node remembers only the next hop and not the entire route, as in source routing.

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

As the RREP propagates back to the source, nodes set up forward pointers to the destination. Once the source node receives the RREP, it may begin to forward data packets to the destination. If the source later receives a RREP containing a greater sequence number or contains the same sequence number with a smaller hop count, it may update its routing information for that destination and begin using the better route.

Routes are maintained as follows: If the source node moves during an active session, it can reinitiate the route discovery procedure to establish a new route to the destination. When either the destination or some intermediate node moves, a special RREP is sent to the affected source nodes. Once the next hop becomes unreachable, the node upstream of the break propagates an unsolicited RREP with a fresh sequence number and infinity hop count to all active upstream neighbors. Those nodes subsequently relay that message to their active neighbors. This process continues until all active source nodes are notified. Upon receiving notification of a broken link, source nodes can restart the discovery process if they still require the destination. Link failure can be detected by using Hello messages or by using link-layer acknowledgements (LLACKS), it also can be indicated if attempts to forward a packet to the next hop fail.

The main benefit of AODV over DSR is that the source route does not need to be

included with each packet. This difference results in a reduction of routing protocol overhead. Because the RREP is forwarded along the path established by the RREQ, AODV requires symmetric links.

2.2.2.4 Signal Stability-Based Adaptive Routing (SSA)

Signal Stability-Based Adaptive Routing (SSA) [20] performs on-demand route discovery by selecting longer-lived routes based on signal strength and location. Selecting the most stable links leads to less route maintenance. Functionally, the SSA protocol consists of two protocols, the Forwarding Protocol (FP) and the Dynamic Routing Protocol (DRP), which utilize the extended device driver interface. This interface is responsible for making available to routing protocols the signal strength information from the device. DRP maintains the routing table by interacting with the DRP on other mobile nodes. FP performs the actual routing table lookup to forward a packet onto the next hop.

Two tables are maintained in the SSA protocol: the Signal Stability Table (SST) and Routing Table (RT). Every node sends out a link layer beacon to its neighbors once every time quantum. Each node classifies its neighbors as strongly connected (SC) or weakly connected (WC) by comparing the received beacon signals strength with a threshold, which are recorded in the SST. Only SC nodes in the SST have an entry in RT, which stores destination and next hop pairs for each known route.

When a source wants to send a packet to the destination, if there is no entry for the destination in the RT, the FP initiates a route search to find a route to the destination by

sending a route search packet. When a node receives the query packet, it propagates the packet further only if the query packet is received over a strong link and the node has not seen this query before (to prevent looping). A query packet, which is received over a weak link, is dropped. When a query reaches the destination, it contains the address of each intermediate node. The destination selects the route recorded in the first received query packet since it probably was received via the shortest path, and then it sends a reply packet back to the source along the selected route. Each intermediate node along the path includes the new (destination, next hop) pair in its RT based on the route contained in the reply packet. If there is no route that consists of strong links, the query packet may never reach the destination. When the source does not receive a reply after some timeout period, it must decide whether it wants to find any route that has strong links or wait and try to find a strong route at a later time.

When a host moves out of range of its neighbors or shuts down, the neighbors will recognize that the node is not reachable because they no longer receive beacons from that node. The DRP will modify the SST and RT to reflect the changes. The node detecting the failure sends an error packet to the source. The source FP will send a message to erase the invalid route, and will also initiate a new route discovery to find an available route.

The advantage of SSA arises from the buffer zone effect. If an SC link is chosen as part of a route, it will have to become WC before breaking, therefore the entire route has a longer life; this in turn reduces the number of route reconstruction required. One of the drawbacks of SSA is that, unlike in AODV and DSR, intermediate nodes can not reply to route requests sent towards the destination, it results in potentially long delays before a route can be discovered. SSA also results in routes with slightly higher hop counts than

optimal routing because of limiting links to strong links. The simulation results in [20] indicate that the hop count is within a factor of 1.1 to 1.5 of the optimal hop count.

2.2.2.5 Flow Oriented Routing Protocol (FORP)

Flow Oriented Routing Protocol (FORP) [9] is a kind of on-demand algorithm; it only maintains routes for "active" source/destination pairs. The difference to other on-demand protocols is that in FORP, the destination predicts the change of topology ahead of time and determines when the flow needs to be routed or "handed off" based on the mobility information contained in the data packets. Though only real-time flow is discussed, the concept also can be applied to non real-time data.

The assumption is made that a given mobile node will be able to predict the link disconnection time of any of its one-hop neighbors (e.g. the time that they move out of transmission range). When a source needs to send a flow to the destination, it will first check its routing table to see if it has an unexpired route to the destination. If not, it broadcasts a Flow-REQ message. This message is processed similarly to the route request message in DSR. When a node is forwarding the Flow-REQ message, it also appends its own ID and the Link Expiration Time (LET) for the last link of the message. When a Flow-REQ message arrives at the destination, it contains the list of nodes along the route it has traveled and the LETs for each hop along the route. The destination can then determine the Route Expiration Time (RET) by using the minimum of the set of the LETs along the route. If the received route is more stable than the one currently in use, the destination sends a Flow-SETUP message back to the source along the chosen route. The

intermediate nodes set up the flow states when they receive the Flow-SETUP message.

During the connection, intermediate nodes append LETs to each data packet; so the destination continues to receive RET predictions from data packets. When the destination has determined that the route is about to expire, a Flow-HANDOFF message is generated and propagated via flooding in the same way as the flow-REQ message. After the source receives a Flow-HANDOFF message, it determines the best route on which to handoff the flow based on the information contained in the Flow-HANDOFF message. The source then sends a Flow-SETUP message, which is the same as the one used before, along the new route. FORP requires each mobile node to know its own position, which can be accomplished through GPS to get its speed and moving direction.

Simulations have been performed to compare FORP with DSDV and LMR (Lightweight Mobile Routing) with 100 mobile nodes in a 500x500m square area [9]. Simulation results show:

- FORP has the highest throughput overall compared to LMR and DSDV. When mobility is lower, the throughput of FORP is slightly lower than at higher mobility, because the initial route is sub-optimal.
- FORP's delay is moderate. But at very low mobility rate, FORP's delay is higher than in LMR and DSDV.
- DSDV generates a significantly larger amount of overhead compared to LMR and FORP because of its periodically broadcasting of routing updates. The overhead of LMR is about 4 times of FORP.

2.3 Other Protocols and Algorithms

In this thesis, we are concerned with the prediction of link availability in ad hoc networks. Besides SSA and FORP discussed above, there are several related protocols and algorithms on link state prediction. Associativity based routing (ABR) [7] favors routes with long-lived links. The proximity model of McDonald et al. [21] proposes an adaptive learning strategy to discover with high probability when two nodes are effectively moving together, to enhance the performance of routing algorithms and better facilitate mobility-adaptive dynamic clustering in ad-hoc networks. He et al. proposes a link state prediction model based on the knowledge of mobile nodes' position [8]. Marti et al. propose two techniques, watchdog and pathrater, that improve the throughput in an ad hoc network in the presence of nodes that agree to forward packets but fail to do so [22]. The watchdog identifies misbehaving nodes, while the pathrater avoids routing packets through these nodes.

2.4 Discussion

2.4.1 Table-Driven vs. On-Demand Routing Protocols

As discussed earlier, table-driven routing relies on a routing table update mechanism that involves the constant propagation of routing information, which incurs substantial signaling traffic and power consumption. Since both bandwidth and battery power are scare resources in mobile computers, this becomes a serious limitation. In on-demand routing, when a route to a new destination is needed, it will have to wait until a route is discovered, but in table-driven protocols, a route to every node is always available. Table 2.1 lists some basic differences between the two classes of protocols [12]. Table 2.2 is a

comparison of several on-demand routing protocols [12].

Table 2.1Comparison of On-Demand versus Table-Driven Based Routing Protocols

Parameters	On-Demand	Table-Driven
Availability of Routing Information	Available when needed	Always available regardless of need
Routing Philosophy[1]	Flat	Mostly flat except for CSGR
Periodic Route Updates	Not required	Yes
Coping with Mobility	Using localized route discovery	Inform other nodes to achieve consistent routing table
Signaling Traffic Generated[2]	Grows with increasing mobility of active routes	Greater than that of on-demand routing
Quality of Service Support	Few can support QoS	Mainly shortest path as QoS metric

1. Addressing scheme: flat or hierarchical
2. Routing message overhead

Table 2.2 Comparison of the Characteristics of On-Demand Routing Protocols

Performance Parameters	AODV	DSR	TORA	SSA
Routing Philosophy	Flat	Flat	Flat	Flat
Loop Free	Yes	Yes	Yes	Yes
Multicast Capability	Yes	No	No	No
Beacon Requirements	No	No	No	Yes
Multiple Route Possibilities	No	Yes	Yes	No
Routes Maintained in	Route table	Route cache	Route table	Route table
Utilizes Route Cache/Table Expiration Timers	Yes	No	No	No
Route Reconfiguration Methodology	Erase route; Notify source	Erase route; Notify source	Link reversal; Route repair	Erase route; Notify source
Routing Metric	Freshest & Shortest path	Shortest path	Shortest path	Associativety & Stability

2.4.2 Comparisons of Simulation Results

Simulation results for some existing ad hoc routing protocols (AODV, DSDV, DSR, TORA) have been reported in numerous papers [10][11][12]. Three metrics are chosen to evaluate these protocols:

- Packet delivery ratio: ratio between the number of packets originated by the application layer source and the number of packets received by the destination. For DSR and AODV, packet delivery ratio is independent of traffic load, with both protocols delivering between 95% and 100% of packets in all cases (with 64 bytes

data packet size). At high rates of mobility, DSDV does poorly, dropping to a 70% packet delivery ratio; nearly all of the dropped packets are lost. TORA delivers over 90% of the packets with 10 or 20 sources. At 30 sources, the network was unable to handle all of the traffic generated by the routing protocol and a significant fraction of the data packets were dropped [11].

- Routing overhead: the total number of routing packets during the simulation. TORA, DSR, AODV are all on-demand protocols. When the number of data sending resources increases, the number of routing packets will increase because there are more destinations to which the network must maintain working routes. However, AODV requires about 5 times the overhead of DSR when there is constant node motion. DSDV has approximately constant overhead, regardless of movement rate or traffic load [10]. TORA's overhead is the sum of a constant mobility-independent overhead, which is the result of IMEP's neighbor discovery mechanism, and a variable mobility-dependent overhead. The simulation results show TORA has the highest routing overhead among the routing protocols mentioned above [10].
- Path optimality: the difference between the number of hops a packet took to reach its destination and the length of the optimal path that physically exists through the network when the packet was originated. Both DSDV and DSR use routes very close to optimal. TORA and AODV each take up more hops longer than optimal for some packets. The length of routes with TORA and AODV each shows significant difference with respect to node mobility, and DSDV and DSR do not [10].

According to the simulation results, AODV and DSR are two ad hoc routing protocols with overall better performance in terms of these three metrics. In the situation

where smaller number of nodes and lower load and/or mobility, DSR outperforms AODV; otherwise, AODV outperforms DSR [11]. DSR has the lowest routing overhead. Because DSR places a source route header in each packet, DSR becomes more expensive than AODV except at higher rates of mobility.

2.4.3 Open Problems

In most of the current ad hoc routing protocols like DSDV, DSR, TORA, AODV etc., a node will keep using the route until the link is broken, then a new route discovery will be used to find a new route to the destination. During this time, the packets will be lost, and it will cause significant throughput degradation. SSA utilizes the link state information, and chooses a long-lived route to reduce route reconstruction, but it still has to find a new route after a link is broken. FORP is the only protocol currently published that uses mobility prediction. It handoffs when a link is predicted to be broken. The simulation results show that it has higher throughput, acceptable delay and lower routing overhead than LMR and DSDV. On the other hand, FORP does not have a mechanism to handle prediction failure, for example an intermediate node of the route suddenly turns off. Also there is no simulation comparison among FORP, DSR and AODV. The last two have much better performance than DSDV and LMR, so the advantage of FORP is not clearly demonstrated.

Chapter 3 AODV and DSR Routing Protocol Simulations in NS2

Before we work on the link state prediction to improve an ad hoc network routing protocol' performance, more detailed information should be obtained. Besides the performance parameters like packet delivery ratio we can acquire, the simulation results will help us trace all the packets (including data packets and routing control messages) transmitted by every mobile node, to find out the situations in which data packets drop. Also, the simulation results based on the original protocols provide us a baseline for comparison purpose with when we implement our prediction algorithm later.

According to the published simulation results, DSDV has approximately constant routing overhead. At high rate of mobility, its packet delivery ratio drops to 70% [10]; TORA has the highest routing overhead, and in a 30 sources case, it essentially underwent congestive collapse. The worst packet delivery ratio drops to 40% [10]. AODV and DSR have high packet delivery ratio and relatively low overhead [10][11][12]. Our focus is therefore on these two prominent on-demand routing protocols: AODV and DSR. Due to time constraints, only the DSR protocol will be modified in this thesis.

3.1 The Network Simulator (NS2)

The simulation tool we choose will be used for simulating the ability of the AODV and DSR protocols to adapt to different scenarios, for analyzing the simulation results, and for further modification of the protocols. The functional requirements for the simulation

tool are:

- Extensible: the simulation tool must be easy to extend if we are to add new functionality.
- Scenario generation: the simulation tool can automatically create complex traffic patterns, topologies and dynamic events.
- Protocol availability: AODV and DSR protocols and related protocols must have been implemented in the simulation tools, so we do not have to build them ourselves.

OPNET Modeler from OPNET Technologies Inc. [23] is a network development environment tool with object-oriented modeling approach and graphical editors. Its Model Library has dozens of models of network protocols, technologies and applications including Wireless LANS (IEEE 802.11). But the ad hoc network routing protocol implementations were not available when this work was started.

The Network Simulator (NS2) is a discrete event simulator developed by the University of California at Berkeley and the VINT project [24]. It provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. The Monarch research group at Carnegie-Mellon University developed support for simulation of multihop wireless networks complete with physical, data link, and medium access control (MAC) layer models on NS2 [10]. It is open source, and can be downloaded from the Internet [25]. It provides tools for generating data traffic and mobile node mobility scenario patterns for the simulation. Also four ad hoc network routing protocols (AODV, DSDV, DSR and TORA) have been implemented. NS2 provides a split-programming model. The simulation kernel is implemented with a systems language (C++), Tcl scripting language is used to express

the definition, configuration and the control of the simulation. This split-programming approach can benefit the research productivity. Also, NS2 can produce a detailed trace file and an animation file for each ad hoc network simulation that is very convenient for analyzing the routing behavior. The disadvantage of NS2 is that it is a large system with a relatively steep initial learning curve.

In NS2, the Distributed Coordination Function (DCF) of IEEE 802.11 for wireless LANs is used as the MAC layer protocol. The radio model uses characteristics similar to a commercial radio interface, Lucent's WaveLAN [26]. WaveLAN is modeled as a shared-media radio with nominal bit rate of 2Mb/s and a nominal radio range of 250 meters. The signal propagation model combines both a free space propagation model and a two-ray ground reflection model. When a transmitter is within the reference distance of the receiver, the free space model, where the signal attenuates as $1/r^2$, is used. Outside of this distance, the ground reflection model, where the signal falls off as $1/r^4$, is used.

A send buffer of 64 bytes is maintained for the AODV and DSR protocols. It contains all data packets waiting for a route. Packets are dropped if they wait in the send buffer for more than 30 seconds. All the packets (data and routing) sent by the routing layer are queued at the interface queue until the MAC layer can transmit them. The interface queue is a priority queue with a maximum size of 50 packets. The routing packets have higher priority than data packets. Here is a summary for the implementation of wireless networks in NS2:

- Mac Layer: IEEE 802.11
- Address Resolution Protocol (ARP)
- Ad hoc routing protocols: DSDV, DSR, TORA, AODV

- Radio Propagation Model:
  - Friss-space attenuation at near distances
  - Two-ray ground at far distances
- Antenna: an omni-directional antenna having unity gain

3.2 The Simulation Model

The overall goal of our simulation experiments is to measure the ability of ad hoc routing protocols to react to network topology changes while continuing to successfully deliver data packets to their destinations. To measure this ability, a variety of workloads were applied to the simulated network, including node movement and data traffic patterns. The parameters selected in this thesis are mostly from the MobiCom'98 paper [10], which are also used by papers published later [11][12].

3.2.1 Movement Space

The ad hoc networks used in our simulation consist of 50 wireless nodes, moving in a rectangular (1500m x 300m) flat space. Because the nominal transmission range is 250 meters for each radio in NS2, choosing a rectangular space can force the use of longer routes between source and destination nodes than using a square space [10].

3.2.2 Movement Model

NS2 can generate different node movement scenario files based on the "random waypoint" model [5]. This model is characterized by two parameters: "pause time" and "maximum speed". Each node begins the simulation by remaining stationary for "pause time" seconds, and then it moves to a randomly selected destination in the

1500m x 300m area at a speed distributed uniformly between 0 and “maximum speed”. When it reaches the destination, the node stops moving again for “pause time” seconds, randomly selects another destination, then repeats the procedures as previously described for the duration of the simulation.

In our simulation, the movement scenario files are generated for 7 different pause time: 0, 30, 60, 120, 300, 600 and 900 seconds. We use two different maximum speeds of node movement: 20 meters per second and 1 meter per second.

3.2.3 Communication Model

We choose the traffic sources to be constant bit rate (CBR) sources. Two data sending rates are simulated: 4 packets per second and 12 packets per second. Two different communication patterns are chosen corresponding to 20 and 30 traffic sources.

The reason why we do not choose TCP sources is that TCP adapts to the load of the network. For the same data traffic and node movement scenario, the time when a node sends a packet will be different if TCP is used in the simulation for simulations using different protocols or different versions of the same protocol. Then it is difficult to compare the performance between different protocols and between the original and modified protocols.

3.2.4 Simulation Time and Cases

In NS2 simulations of the ad hoc networks, all communication patterns are peer-to-peer, and connections begin at times uniformly distributed between 0 and 180 seconds. So the simulation time must be greater than 180 seconds. To choose an appropriate simulation time, we ran four simulations (2 for DSR and 2 for AODV) with 0 second pause time and

maximum speed of 20 meters per second, the longest simulation time is 1000 seconds. At 100 seconds intervals, we ran 10 simulations for each scenario. Figure 3.1 and Figure 3.2 are the results of the simulation. For the meaning of the packet delivery ratio and normalized routing load, please see 3.2.5.

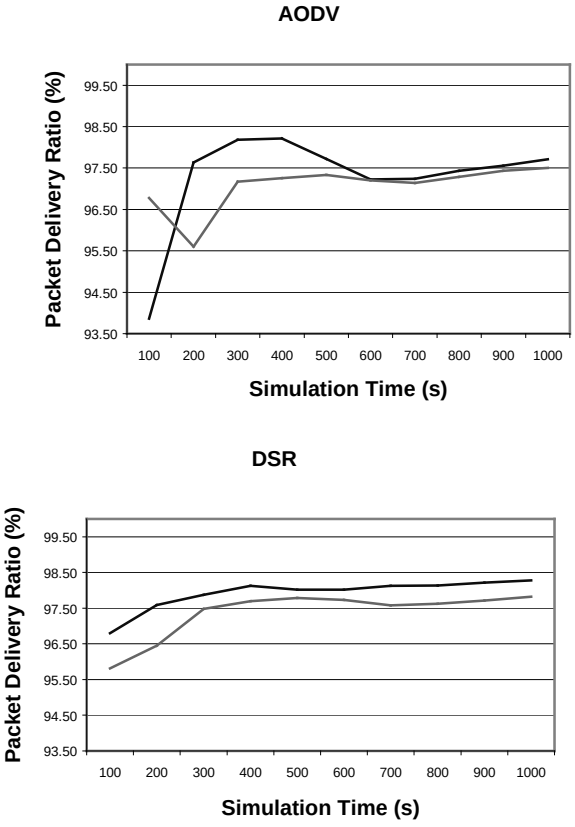


Figure 3.1 Packet Delivery Ratios as Function of Simulation Time for AODV & DSR

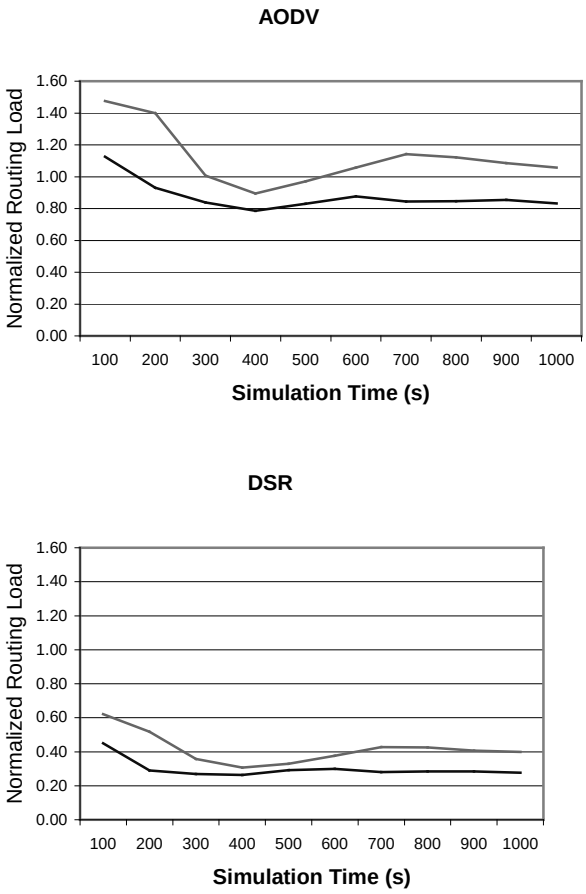


Figure 3.2 Normalized Routing Loads as Fuction of Simulation Time for AODV & DSR

From Figure 3.1 and Figure 3.2 we can see that after 650 seconds of simulation, the packet delivery ratio and normalized routing load are fairly stable. Getting more dropped packets because of longer simulation time will help us analyze and compare protocols. On the other hand, a longer simulation time will cause much longer machine running time and larger simulation trace files, so we choose 900 seconds simulation time. 0 second pause time corresponds to continuous motion; 900 seconds pause time corresponds to no motion during the simulation time.

Because the performance of the protocols is very sensitive to movement patterns, for

each pause time we generate 7 different scenario files. Now we have two different traffic sources, two different maximum speeds, seven different pause times, and two different protocols. Each of them has seven different scenario files; the total is 2x2x7x2x7=392 simulations. Considering that a single simulation takes up to 40 minutes in a Pentium II PC, seven different scenario files for each pause time is reasonable for our simulations.

3.2.5 Performance Metrics

To evaluate the performance of the routing protocols, the following three metrics are used:

- Packet delivery ratio: Because we are more concerned with the percentage of dropped data packets, the packet delivery ratio is defined as the percentage of total not dropped CBR packets, i.e. (total generated CBR packets - total dropped data packets)/total generated CBR packets. In some papers [10] [11] it is defined as the ratio between the numbers of packets originated by CBR sources and the number of packets received by the CBR sink at the final destination. The difference between the two definitions is whether we count the data packets in the queue in the end of the simulation. We find that this number is very small.
- Normalized routing load: The number of routing packets transmitted per data packet delivered at the destination. Each hop-wise transmission of a routing packet is counted as one transmission. It is a better metric than absolute number of routing packets, because it shows the relationship between routing packets and CBR packets.
- Hop count ratio: The ratio between the total number of hop all packets travel to reach

their destinations and the total length of the shortest path that physically existed through the network when the packets are originated. NS2 will provide the length of the shortest possible path between all nodes in the network.

The packet delivery ratio is a very important metric because it describes the loss rate. The routing load metric represents the efficiency and scalability of the routing protocol. Protocols with many routing packets will increase the probability of packet collisions and may delay data packets in network interface transmission queues because routing packets have higher priority. The hop count ratio measures the protocol's ability to use network resource efficiently by finding the shortest path from source to destination node.

3.3 DSR and AODV Simulation Results

We conducted simulations using two different node movement speeds: a maximum speed of 20 meters per second and 1 meter per second. First we compare results that use 20 meters per second. The AODV and DSR simulations share the same data traffic and mobility scenario files in order to allow for a fair comparison. A Perl program was written to read the simulation trace files and extract the metrics we need. All simulations are done with NS2 version 2.1b6 on Linux.

3.3.1 Packet Delivery Ratio

Figure 3.3 shows the packet delivery ratio of DSR and AODV, as a function of both node mobility rate (pause time) and network load (number of data sources).

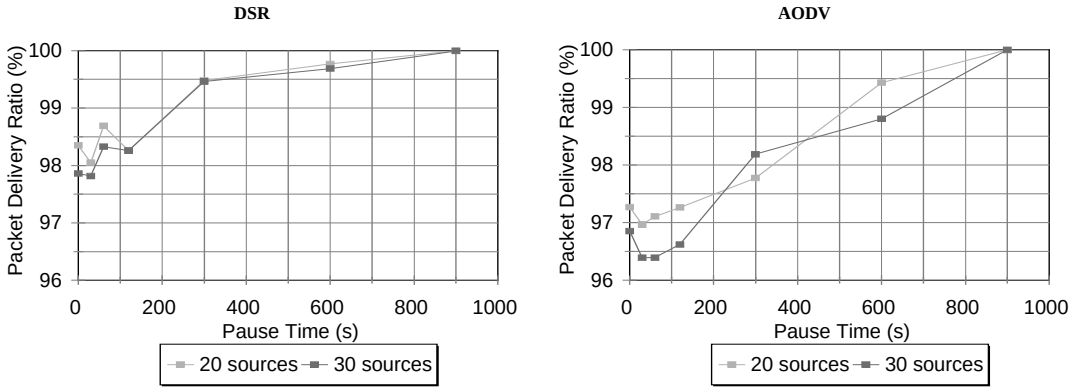


Figure 3.3 Packet Delivery Ratios as Function of Pause Time (Maximum Speed 20m/s)

For AODV and DSR, packet delivery ratio is independent of the traffic load, with both protocols delivering between 96% to 100% of the data packets in all cases. To test the ability of handling high traffic load for both protocols, we ran simulations where the data rate was increased to 12 packets per second from 4 packets per second. The number of data sources is 20; maximum speed is up to 20 meters per second. Figure 3.4 shows the simulation results. The packet delivery ratio dropped significantly for both protocols compared to the lower data rate. Even in the stationary (900 seconds pause time) scenario, it only reached about 95%.

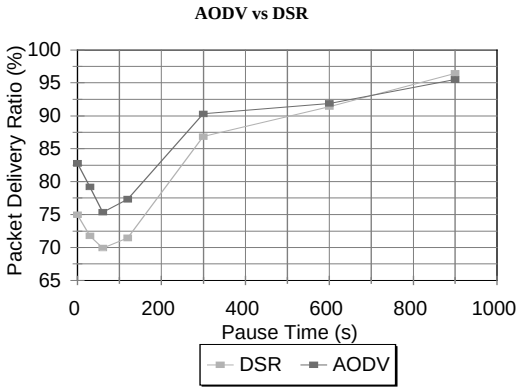
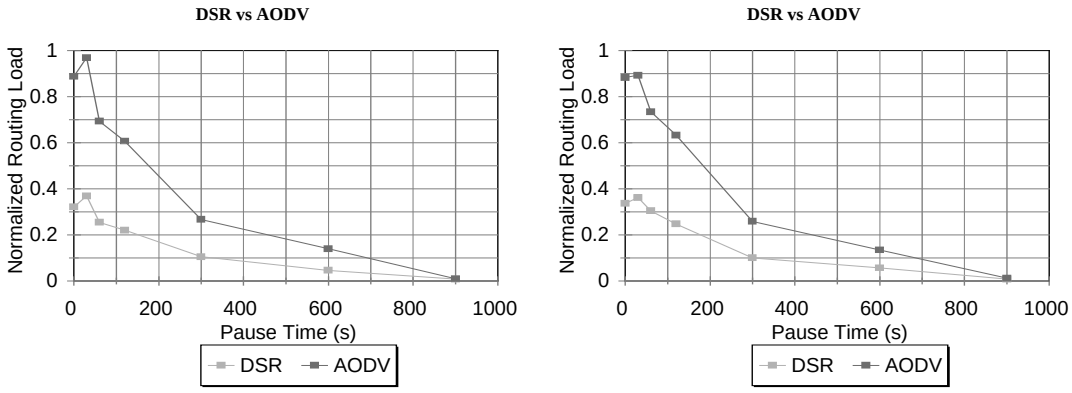


Figure 3.4 Packet Delivery Ratio for 12 Packets per Second

3.3.2 Normalized Routing Load

Figure 3.5 shows the normalized routing load of both protocols in obtaining the delivery ratio shown in Figure 3.3.



(a) 20 sources (b) 30 sources

Figure 3.5 Normalized Routing Loads as Funtion of Pause Time

Because AODV and DSR are on-demand routing protocols, they have almost identically shaped curves. As the node mobility or the number of data sources increases, the routing packets will increase because the routes are more likely to be broken or more working routes must be maintained. For 20 and 30 sources, both protocols have similar normalized routing load.

Figure 3.5 shows that the absolute overhead induced by AODV and DSR are very different. When the nodes are in constant motion (pause time is 0 second), AODV requires about 3 times the overhead of DSR in terms of control messages. In AODV, the route discovery packets typically propagate to every node in the ad hoc network. DSR reduces the scope and overhead of Route Request packets by replying from a node's route cache if it can and uses non-propagating Route Requests.

Figure 3.6 shows the normalized routing load with 12 packets per second,

corresponding to Figure 3.4. Comparing with Figure 3.5 (a), that also uses 20 data sources, the values in Figure 3.6 for both AODV and DSR are much lower. The main reason is more data packets are sent. The absolute number of routing packets is little more in the later case.

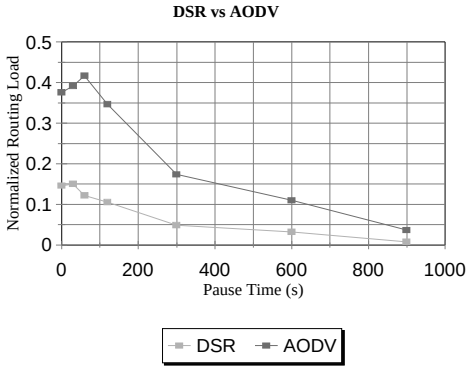
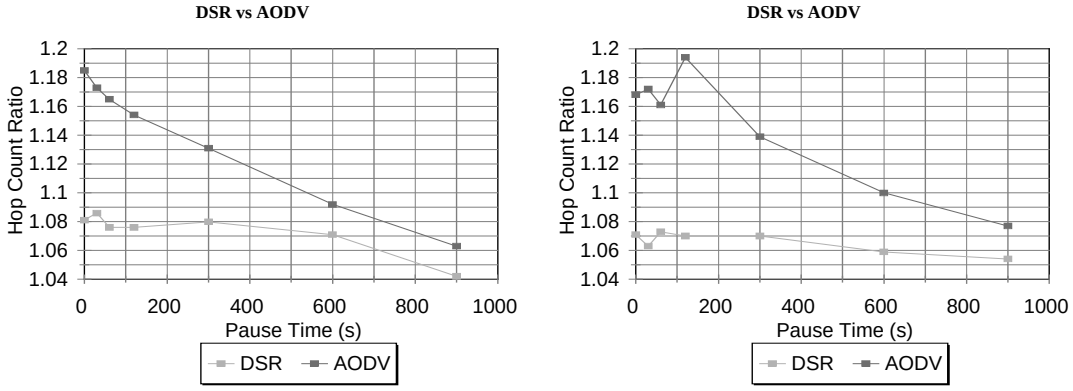


Figure 3.6 Normalized Routing Load as Function of Pause Time (12 Packets/s)

3.3.3 Hop Count Ratio

Figure 3.7 shows hop count ratio for AODV and DSR.



(a) 20 sources (b) 30 sources

Figure 3.7 Hop Count Ratios as Function of Pause Time

With both 20 and 30 sources, DSR has routes that are closer to the shortest path than AODV. Also DSR has no significant change with respect to pause time, on the other

hand, AODV normally has lower hop count ratio while pause time increases.

3.3.4 Low Speed of Node Movement

Figure 3.8 shows packet delivery ratio and Figure 3.9 shows normalized routing load for AODV and DSR with 20 and 30 sources respectively.

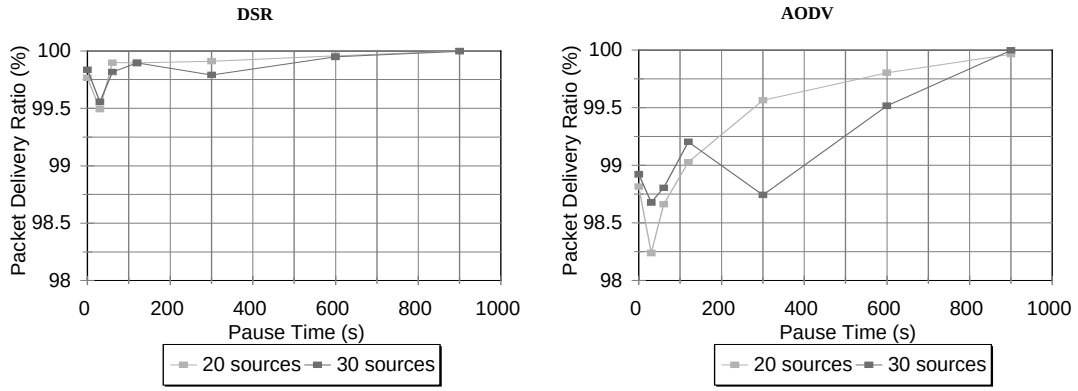


Figure 3.8 Packet Delivery Delivery Ratios as Function of Pause Time (Maximum 1 m/s)

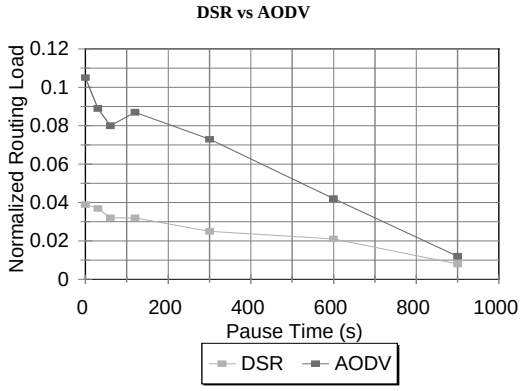


Figure 3.9 Normalizd Routing Loads as Function of Pause Time (Maximum 1m /s)

Both protocols exhibit excellent performance at this movement speed. DSR delivers more than 99.5%, and AODV delivers 98.5% of total data packets. Also the normalized

routing load has been reduced for both protocols because the route may be kept longer at lower speed. But there is still about 2.5 times more overhead produced by AODV than DSR in the highest mobility scenario (0 second pause time).

3.4 Observations and Discussions

The simulation results bring out several important characteristic differences in the two on-demand routing protocols.

- Packet Delivery Ratio: in 4 data packets per second situations, DSR and AODV both have very good performance at all scenarios, and DSR outperforms AODV. In more "stressful" scenarios (12 data packets per second with maximum speed up to 20m/s), AODV outperforms DSR. This conclusion is confirmed by other published simulation results [11] that use 512-byte data packet size. The poor performance of DSR in high "stressful" scenarios is mainly attributed to its aggressive use of route caches, and the lack of any mechanism to expire stale routes or determine the freshness of routes when multiple alternative routes are available.

- Normalized Routing Load: in all scenarios, DSR has much lower routing load than AODV in terms of packet counts (by a factor of up to 3). First, DSR uses route caches to optimize the Route Discovery procedure; second, AODV's overhead is dominated by RREQ packets (often as much as 90%), DSR is dominated by RREP and RERR packets, but there are significantly fewer RREQ packets than in AODV [11].
- Hop Count Ratio: In all scenarios, DSR uses routes much closer to the shortest paths than AODV. The simulation results also suggest that in lower mobility, both protocols can discover routes close to the optimal paths.

- Comparing the simulation results with the results published in [10], most of them match well except the routing overhead. The absolute number of routing packets for the DSR protocol are almost the same, but for AODV, the routing overhead in our simulations are up to one third less than the results in the paper. The explanation for the difference is the improvement in the AODV protocol implementation since the paper was published. According to the AODV source code in NS2, the implementation has been optimized. Also, the simulation results published by the AODV protocol authors in other papers show that the routing overhead is reduced in AODV, but the routing overhead for DSR stays the same [11][12].

3.5 Conclusion

The AODV and DSR protocols simulation results show that both have excellent packet delivery ration except in high “stress’ situation (high data rate, more sources). Their routing overhead decreases as node mobility is getting lower, and DSR always has lower routing overhead (up to 3 times) than AODV in terms of number of packets. DSR can make use of network resource well because its route length (in hop count) is much closer to the shortest path than AODV. Also the simulations provide hands-on experience for the analysis and baseline data for further comparison.

Our plan was to modify both AODV and DSR using our link status prediction algorithm. Due to time limit, it was possible to work on only one protocol in this thesis though. Because DSR induces fewer overheads, and its lack of a mechanism to remove stale routes in the route cache, it is expected to benefit more from pro-active route maintenance, so we will focus more on DSR. The simulation results of both

protocols provide useful reference for further work.

Chapter 4 Link Prediction Algorithm and Simulation

4.1 Prediction Algorithm

There are basically two ways to predict the connectivity between two neighboring nodes. In the first method, we assume a free-space propagation model [27], where the received signal strength of a mobile node solely depends on its distance to the transmitter. Therefore, if we know the motion parameters of two neighbors (e.g. speed, direction, and transmission range), we will be able to determine the duration of time that these two mobile nodes will remain connected. A simple calculation model is [8]: suppose from time t_0 to t, node A and node B do not change their speeds and directions, which means that v_A, v_B and θ in the Figure 4.1 are constants against time t, l and m are fixed. Let t_0 be the starting point. We can calculate the distance d between node A and node B at time t, which can be expressed as a function of t [8]:

$$d^2 = (l + v_A t)^2 + (m + v_B t)^2 - 2\cos\theta (l + v_A t)(m + v_B t)$$

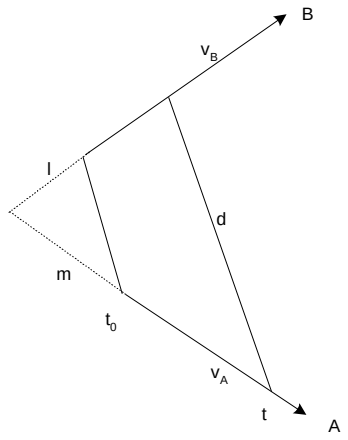


Figure 4.1 Schematic for Prediction Model Using GPS

If we know the transmission range d, we can predict the time t when mobile node A and node B will loose connectivity. This method will require each mobile node to know its own position, speed and heading. This can be accomplished through such sources as Global Positioning System (GPS).

The second method to predict connection failure time uses received signal power measurements. This method has been proposed in [28]. Basically, assume that the sender power level is constant. Received signal power samples are measured from packets received from a mobile node's neighbor. From this information it is possible to compute the rate of change for a particular neighbor's signal power level. Because the signal power threshold for the wireless network interface is fixed, the time when the power level drops below the acceptable value can be computed.

4.1.1 Radio Propagation Models

In our prediction algorithm, an optimistic radio transmission model is used. Node A has a transmission range with radius R. If node B is located within this circle, it is assumed to correctly receive node A's transmission. So the link availability of two mobile nodes can be simply determined by the distance between them.

According to the NS2 implementation, Two Ray Ground Reflection Approximation is used as radio propagation model [29]. At near distances, Friss free-space attenuation ($1/r^2$) is used, at far distance, an approximation to Two Ray Ground ($1/r^4$). The approximation assumes specula reflection off a flat ground plane. The crossover point is called the reference distance. In the NS2 implementation, the reference distance is about 86.14 meters (corresponding to a signal power of 2.59×10^{-8} W at the receiver

node according to NS2's parameters). Because we are concerned with mobile nodes that are going to move out of the senders' transmission range, the prediction algorithm is based on Two Ray Ground Reflection model, only signals below this value will be used for the algorithm. It can save CPU time by ignoring many signal power values when two nodes are still close to each other. It also guarantees that only the Two Ray Ground model is used in the algorithm.

According to the NS2 implementation, the Two Ray Ground Reflection equation is:

$$P = \frac{Pt * Gt * Gr * (ht^2 * hr^2)}{d^4} \qquad (1)$$

Where: P is the signal power at receiver.

Pt is the signal power at transmitter.

Gt is the gain for a signal to a node from the transmitter.

Gr is the gain for a signal to a node from the receiver.

ht is height of transmitter antenna.

hr is height of receiver antenna

d is the distance between transmitter and receiver.

We can assume Pt is a constant. Also, in our wireless ad hoc network simulation, a directional antenna is used. Further, we assume the ground is flat, and that hr and ht are constants. So we can simplify equation (1) under the conditions of ad hoc wireless network simulation:

$$P = k\frac{Pt}{d^4} \qquad (2)$$

Where: $k = Gt * Gr * (ht^2 * hr^2)$ is a constant.

This equation means the signal power at receiver node has relation ($1/d^4$) with the

distance between the sender node and receiver node.

4.1.2 Node Movement

As described in Chapter 3, in the NS2 movement model, a mobile node will move at a constant speed randomly selected up to a given max value to a random destination, then stops for a predefined pause time, and moves again. Let node B be the receiver node, node A be a sender, and also one of node B's neighbors. Node A is moving at speed $\vec{v}_a$, node B is moving at speed $\vec{v}_b$. $\vec{v}_a$ can be decomposed to $\vec{v}_{ax}$ and $\vec{v}_{ay}$, same for $\vec{v}_b$, see Figure 4.2.

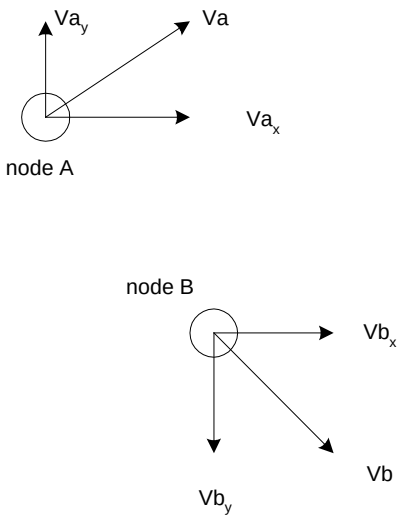


Figure 4.2 Mobile Nodes Moving at Random Speeds and Directions

Then the relative speed of node B to node A is:

$$\vec{v} = (\vec{v}_{bx} - \vec{v}_{ax}) + (\vec{v}_{by} - \vec{v}_{ay}) \qquad (3)$$

According to equation 3), we can assume that relative to node B, node A is still, node B is moving and at relative speed and direction $\vec{v}$.

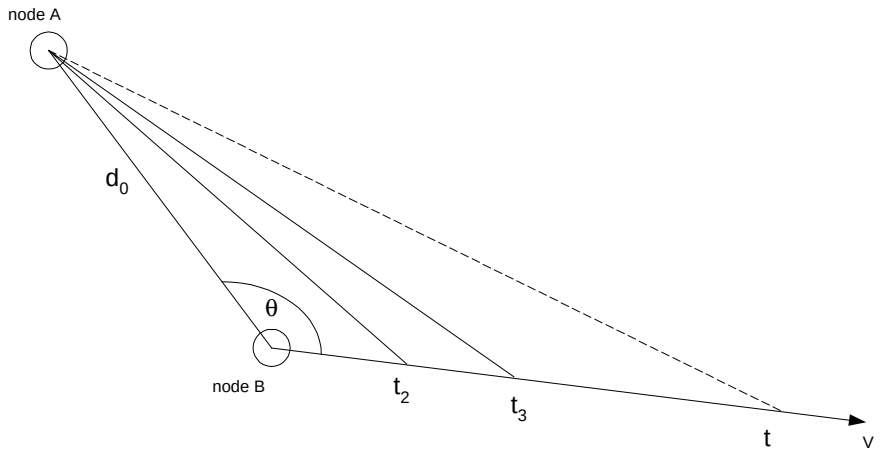


Figure 4.3 Relative Movements of Two Mobile Nodes

4.1.3 Prediction of Critical Time

The critical time is defined as the time when two nodes are moving out of the radio transmission rage. For a wireless network interface, because the signal power threshold is fixed, when two nodes maintain their moving speed and direction, the critical time is a constant. As illustrated in Figure 4.3, at time T_1, node B received a signal from node A, assumes the distance between them is d_0, then:

$$P_1 = k\frac{P_t}{d_0^4} \qquad (4)$$

At time T_2, node B received the second signal from node A, let $t_2=T_2-T_1$,

$$P_2 = k\frac{P_t}{(d_0^{\,2} + (vt_2)^2 - 2d_0 vt_2\cos\theta)^2} \qquad (5)$$

At time T_3, node B received the third signal from node A, let $t_3=T_3-T_1$, and t_3 is not necessarily $n*t_2$, n is an integer. The spacing in time is arbitrary.

$$P_3 = k\frac{P_t}{(d_0{}^2 + (vt_3)^2 - 2d_0 vt_3\cos\theta)^2} \qquad (6)$$

At time T, the node B will receive a signal with power equivalent to the threshold Ps, let t=T- T_1, also assuming during time T_1 to T nodes A and B maintain their speed and direction:

$$P_s = k\frac{P_t}{(d_0{}^2 + (vt)^2 - 2d_0 vt\cos\theta)^2} \qquad (7)$$

From equation (4), we can get:

$$P_1 d_0{}^4 = kP_t \qquad (8)$$

Substituting (8) into equations (5), (6) and (7)

$$\sqrt{P_2} = \frac{\sqrt{P_1}d_0{}^2}{d_0{}^2 + (vt_2)^2 - 2d_0 vt_2\cos\theta} \qquad (9)$$

$$\sqrt{P_3} = \frac{\sqrt{P_1}d_0{}^2}{d_0{}^2 + (vt_3)^2 - 2d_0 vt_3\cos\theta} \qquad (10)$$

$$\sqrt{P_s} = \frac{\sqrt{P_1}d_0{}^2}{d_0{}^2 + (vt)^2 - 2d_0 vt\cos\theta} \qquad (11)$$

From (9) and (10) we can get:

$$v^2 = \beta d_0{}^2 \qquad (12)$$

Where: $\beta = \dfrac{(\sqrt{P_1P_2}t_2 + \sqrt{P_2P_3}t_3 - \sqrt{P_1P_3}t_3 - \sqrt{P_2P_3}t_2)}{(t_2t_3{}^2 - t_3t_2{}^2)\sqrt{P_2P_3}}$ is a constant.

From (9) and (11) we can get:

$$\sqrt{P_s} = \frac{\sqrt{P_1 P_2}\, d_0{}^2 t_2}{(t_2\sqrt{P_2} - t\sqrt{P_2} + t\sqrt{P_1}) d_0{}^2 + (t_2\sqrt{P_2}\, t^2 - t_2{}^2\sqrt{P_2}\, t) v^2} \quad (13)$$

Substituting (12) into (13):

$$\sqrt{P_s} = \frac{\sqrt{P_1 P_2}\, t_2}{(t_2\sqrt{P_2} - \sqrt{P_2}\, t + \sqrt{P_1}\, t) + (\sqrt{P_2}\, t_2 t^2 - \sqrt{P_2}\, t_2{}^2 t)\beta} \quad (14)$$

Then we obtain the equation:

$$at^2 + bt + c = 0 \quad (15)$$

Where: $a = t_2\sqrt{P_2 P_s}\,\beta$

$$b = \sqrt{P_s}\,((\sqrt{P_1} - \sqrt{P_2}) - t_2{}^2\sqrt{P_2}\,\beta)$$
$$c = t_2\sqrt{P_2 P_s} - t_2\sqrt{P_1 P_2}$$

Finally the critical time is:

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (16)$$

Because t cannot be a negative value, then:

$$t = \frac{\sqrt{b^2 - 4ac} - b}{2a} \quad (17)$$

In the rest of the thesis, we call the prediction time derived from equation (17) based on three received packets' signal power strength the prediction algorithm.

4.2 Implementation of the Prediction Algorithm

From the prediction algorithm, at least three packets are needed to correctly predict future

disconnection to the neighboring node. In the implementation, each mobile node will keep an array (called signalInfo array) of signal info objects. Each signalInfo object holds three packets with information such as signal power strength and reception time for the same neighboring mobile nodes. When node B receives packets from node A, it updates its signalInfo array according to:

$$P_3 \leq P_2 \leq P_1 \text{ and } T_3 > T_2 > T_1 \quad (18)$$

When two mobile nodes are moving closer, the latest signal power strength will be greater than the previous one. In this case, we set P_1 to the latest signal power value and set P_2 and P_3 to zero, no prediction is necessary.

Some comments about the implementation:

- We could make use of two packets to make prediction by simply assuming that signal power decreases linearly. In the simulation later, we found that this is not accurate and always underestimates the real link breakage time, which causes many "false predictions". Therefore we will not use signal power strength from only two packets to make predictions in the rest of the thesis.
- This algorithm assumes that during T_1 and critical time T, two nodes maintain their speed and direction. In reality, two nodes may change their speed or/and their direction during this time. According to the algorithm, the prediction based on the first one or two packets just after the changes may not be correct and cause a "false prediction". For example, when the relative speed of two nodes moving away slows down, the first two packets node B received match the condition (18) above, but the prediction based on these three packets will not be correct.
- This algorithm assumes there is no noise during the packet transmission, which is also

used in NS2. This is a very simplistic assumption. Some experiments that add randomly produced noise (5% to 10% of signal power strength) to the signal have been conducted. The simulation results show the prediction accuracy suffers because the signal power difference between continuous received packets from the same neighboring node is less than 1%. The purpose of this thesis is reducing dropped packets based of the link prediction by making using of link status information. Signal power provides a simple, efficient and economic way to make prediction. It could be used with GPS/location information in case of random fluctuation to improve the prediction accuracy.

4.3 Simulation Results

All the link predictions are obtained from the prediction algorithm discussed above, which is based on the calculations of received packets signal power strength. These packets are sent by mobile node A, received and computed by mobile node B.

Because the ad hoc network protocol needs a period of time to set up a new route, a time parameter **critical time** t_s is introduced. t_s should be big enough for node B to send a route error message to the source of the packet and the source to find a new route. When node B receives a packet from node A at time T_r, it gets the prediction that the link between node A and node B will be lost at time T_d. If $T_d - T_r \leq t_s$, we say the link enters the **critical state**. Node A will send a routing error message to the source to look for a new route. Another parameter is **window time** t_w, the difference allowed between prediction time and the time a packet is really dropped because the link breaks. Because link break will cause a packet to be dropped, the packet drop time is always later than the

link break time.

Before modifying the ad hoc network protocols, the prediction algorithm was implemented in NS2, and some simulations were done to verify the correctness of the prediction algorithm. In this thesis, node B only monitors unicast packets, for which B is the next hop from node A. To trace the prediction time more easily, two items are added to every line marked ‘r’ (indicating receiving a packet) in the trace file of the simulation. One is the sender node id and the other is the prediction time. When node B receives a unicast packet from node A, if it is the next hop of the packet, it will add it to its signalInfo array and calculate the prediction time. The prediction time will appear in the trace file. Perl scripts are used to search the trace file to find out if and when the link really breaks (indicated by a packet being dropped over this link), and to explore the reasons if no packet dropped. For example, one Perl script is used to find if the packet dropped as predicted. It reads the trace file, compares the packet reception time and the prediction time. If the link enters the critical state, it will search through the trace file to find out if there is a packet dropped within the prediction time on this link or whether this link is not in the current route anymore.

Table 4.1 and Table 4.2 are the simulation results for the AODV and DSR protocols respectively. All the dropped packets listed in the following tables are CBR data packets.

In the simulation, “window time” is set to 1 second. Because in ad hoc networks the critical time will be different for routes with different hop counts and different network topologies, it is difficult to find a value suited for all scenarios. Also, in DSR, if the source node has an alternative route in its route cache, it does not have to start the route request routine; the critical time will be much shorter. Based on our analysis of the

simulation trace file, packet transmission time from source to the destination is in the range of dozens of milliseconds. The critical time in the simulation is set to 3 seconds.

Table 4.1 Link Prediction Distribution: DSR Protocol

	20-01	20-301	20-3020	20-020
Total no route dropped packets[1]	66	80	899	839
<1 second since receiving latest packet[2]	34	43	383	287
Dropped in predicted time[3]	34	43	383	272
Dropped but not predicted[4]	0	0	6	15
Dropped out of predicted time[5]	0	0	3	5
Predicted but not dropped				
Route switched packets[6]	80	70	838	984
Dropped packets not found: [7]	0	1	72	43
Moving closer[8]	0	0	23	29
Other reasons[9]	0	1	49	14

Table 4.2 Link Prediction Distribution: AODV Protocol

	20-01	20-301	20-3020	20-020
Total no route dropped packets[10]	110	44	2003	1439
<1 second since receiving latest packet	109	44	992	837
Dropped in predicted time	109	45	957	802
Dropped but not predicted	0	0	36	39
Dropped out of predicted time	0	0	17	1
Predicted but not dropped				
Route switched packets	31	22	572	698
Dropped packets not found:	0	0	63	45
Moving closer	0	0	23	26
Other reasons	0	0	40	19

Notes:

- All simulations are run on NS2 version 2.1b6 in Linux using DSR (Table 4.1) and AODV (Table 4.2) protocols with 50 mobile nodes and 20 data sources, with CBR packets at rate 4 packets per second. The mobile area is 1500 x 300 meters.
- Four mobility patterns are simulated: 0 second and 30 seconds pause time and up

to 1m/second and 20m/second moving speed respectively, resulting in four combinations. In the table, the four cases are:

- 20-01: 0 second pause time, up to 1m/second moving speed.
- 20-301: 30 seconds pause time, up to 1m/second moving speed.
- 20-3020: 30 seconds pause time, up to 20m/second moving speed.
- 20-020: 0 second pause time, up to 20m/second moving speed.

These four cases represent different scenarios, from lowest to highest mobility.

1. Total number of packets dropped because of "no route" error.
2. The time between node B receiving this packet from node A and the previous packet from A is less than 1 second. This parameter shows that the connection between two mobile nodes is in the active mode. Because the data rate is 4 packets per second in the simulation, a link in active mode implies that the node B can obtain enough information to make a correct prediction.
3. The packets really dropped at time $\leq T_d + t_w$. For example, the algorithm predicts that the link will be lost at 100.0 second, if the actual packet drops at 99.8 second or 100.5 second, this packet is counted in this category.
4. The total number of packets dropped that were not predicted. It happens when the packet is the first or second packet transferred between two nodes, because there is not enough information to calculate a prediction time. In this situation, two nodes might not transmit packets before; or they transmitted packets before, but the mobility pattern changed (their speed and/or direction), for example, nodes move apart faster.
5. When the prediction algorithm is verified, we analyze the simulation trace file

from time T_r to time $T_r+ t_s +t_w$. If the dropped packet is found between T_r to time T_d+t_w, that counts as "Dropped in predicted time"; if the packet is dropped between time T_d+t_w to time $T_r+ t_s +t_w$, it counts as "Dropped out of predicted time", which implies a lack of accuracy of the prediction algorithm. It may be caused by movement pattern changes, for example nodes move apart slower.

6. During the time T_r to time $T_r+t_s +t_w$, a new route for the same source/destination pair appeared, and does not include the link between A and B. No packet was dropped over the A-B link. In DSR, this also includes the packets that have been salvaged by node A.
7. No packet is dropped during the prediction time between node A and node B, also no new route appeared.
8. One reason of the "not found" is that at first node A and node B are moving away, but during the prediction time they are moving closer.
9. There are some other reasons for "not found":
    - The prediction calculation is based on the first few packets that may cause the prediction time to be inaccurate.
    - The two nodes are still moving away, but at a slower speed.
10. In AODV simulations, there could be many dropped packets (dozens, even hundreds of packets) on the same link because of "no route" error during a period of time. It seems that the source nodes are not aware of the broken link. In this calculation, these packets are not counted.

4.4 Parameters Setting

The prediction algorithm assumes that two mobile nodes will maintain their speed and direction between the time the receiver received a packet and the prediction time when the packet will drop. In reality, the speed and moving direction of two nodes could change at any time. Two nodes could be moving in parallel, moving closer faster/slower or moving away faster/slower. These changes will affect the prediction correctness, even cause "false prediction", for example at first, two nodes move apart, after entering the critical state they are moving closer. But once two nodes maintain their movement patterns, node B can obtain correct prediction times again after receiving at least three packets.

Obviously, the closer the prediction time and the more time error tolerance allowed, the more accurate the prediction correctness can get. The simulation results for different critical time and window time are listed in Table 4.3, using 20-020 DSR as an example scenario. Other parameters are as before.

Table 4.3 Prediction Distributions for Different Parameters

Parameter (critical time, window time) in second	(3,1)	(2,1)	(1,1)	(1,3)
Dropped in predicted time	272	275	277	281
Dropped out of predicted time	5	3	1	0
Route switched packets	984	926	842	843
Not found	43	31	19	17
Moving closer	29	21	14	13
Other reasons	14	11	5	4

The results of Table 4.3 show that if we reduce critical time from 3 seconds to 1 second, more "false predictions" can be avoided, but few gains will be achieved when we extend the "window time" from 1 second to 3 seconds.

4.5 Discussion

- For active links, more than 95% of the dropped packets are predictable. The other 5% happen when they are the first one or two packets transmitted between nodes A and B. From our simulation results, the total number of active links can be used as a parameter representing the predictable link states.
- In high mobility (20-3020 and 20-020), the active links account for roughly one third of the total number of no route dropped packets for DSR, half for AODV. This is because DSR uses route caches to reply to Route Request messages and some stale routes may be used.
- The prediction algorithm cannot make prediction if there are not enough received packets available. One way to improve the prediction is that all the mobile nodes set their network interface into promiscuous receive mode. All the packets transmitted by other nodes would be monitored if the signals' power strengths are above the threshold. Then more information about neighboring nodes can be obtained even if no packets are directly sent by them. Also, because there is no active transmission between the node and some neighboring nodes, how and whether to inform the neighboring nodes is worth considering. The DSR protocol only maintains active routes. From Table 4.1 and 4.2, the total active links ("less than 1 second" situation) only account for one third to half of the total number of packets dropped because of no route, so there is still more space to improve the prediction ability. But the choice must be made between the possible improvements and the additional overhead caused (computing time, battery power consumption in receiving every packet). In the algorithm implementation, all CBR packets from node A to node B are monitored by

node B to get the prediction time, plus some unicast routing messages (for example, Route Reply and Route Error in DSR) from node A to node B.

- According to the DSR protocol, each mobile node has a route cache that may contain alternative routes to the destination. In the "Route switched" cases, there are several situations. First, the link between node A and node B is actually broken, node A is aware of it, and salvages the packet by switching the route, then sends a Route Error message to inform the source of the link problem. Second, node A and node B both are not in the new route, for example another link in the route also broke before link A-B broke, and the source finds a new route. Third, node A and node B are still in the new route, but node B is not the next hop of node A. After analyzing the simulation trace files, the first scenario (node A salvages the packet) accounts for most of the "Route switched" cases.
- The "moving closer" situation is the worst case for the prediction, because it is a false prediction and will cause overhead. Results of Table 4.3 suggest that if the prediction is made closer to the time when the link is expected to break, the possibility to avoid false prediction will increase.
- The prediction algorithm at least needs three packets to calculate the prediction time. The linear approximation based on two packets caused many "false prediction" because its prediction value is always smaller than the real value.
- The critical time will be different for different network configurations: the number of mobile nodes, the speed of mobile nodes, the topology of the ad hoc network. It will be different even for different routes in the same ad hoc network because of different hop counts. It is difficult to determine dynamically appropriate critical times for each

route in the ad hoc network. Here we use a static critical time for all routes. Determining optimal critical times is a good candidate for future work.

Conclusion: A prediction algorithm based on received packet signal power strength is developed and implemented in NS2. Three packets from an upstream node, while two nodes maintain their movement pattern, are needed to make a correct prediction. The simulation results of NS2 show that the prediction algorithm can predict link failures of active links with above 90% accuracy. Because mobile nodes may change their mobility pattern randomly, the prediction algorithm may make “false predictions” which account for less than 5% of predictable dropped packets. The “false prediction” mostly happens in high mobility scenarios. It will cause overhead, and can be reduced when the implementation parameters are optimized. When the prediction is made closer to the actual link breakage, the more accurate it can be made, but the improvement on predictable link state is not significant. Considering the critical time in the current typical size of simulated ad hoc network, 1 second will be appropriate. There is space to improve the prediction ability if more packets are monitored.

Chapter 5 Pro-active Route Maintenance

After verifying the prediction algorithm in Chapter 4, a pro-active route maintenance mechanism using prediction algorithm was implemented in the DSR protocol (in the rest of the thesis, it is also called DSR with prediction algorithm). The reason that the DSR protocol was chosen for implementing the prediction algorithm is because it has comparable performance as AODV, but also uses snoop, route cache and salvage methods that are more complicated than AODV. It is therefore more of a challenge and a good example to test the prediction algorithm's flexibility.

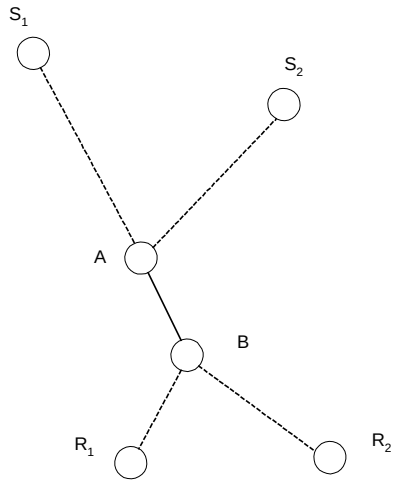


Figure 5.1 Several Routes Share a Link

5.1 Pro-active Route Maintenance in DSR Protocol

Figure 5.1 illustrates some packet transmission paths in an ad hoc network. S_1 and S_2 are sources of the packets. R_1 and R_2 are destinations. Node A and node B are two mobile nodes on the S_1-R_1 and S_2-R_2 paths. So two routes share the A-B link. The dotted line

represents that there may be some intermediate nodes between them. A solid line means that the two nodes at the end of the line are neighbors.

First we review DSR route maintenance briefly. As shown in Figure 5.1, source S_1 sends a packet to destination R_1. Node A and B are intermediate nodes. When node A tries to send a packet to its next hop node B, if A detects the link between node A and B is broken, it will check whether there is an alternative path to the destination R_1 of this packet. If node A can find one, it will use the new route to send the packet to the destination (which is called packet salvaging); otherwise it simply drops the packet. In both situations, node A will send a Route Error message to the source S_1 to inform it that the link A-B is broken. The source will find a new route to send packets if it still has packets to send.

In the original DSR protocol, the source node will keep using one route until the link is broken or Route Reply and Route Error messages are received. The drawback is that the source node will take extra time to find a new route by broadcasting a Route Request message if it cannot find one in its route cache. The packet will be dropped if the upstream node of the broken link cannot salvage this packet. The consequence is that the throughput of the network is reduced.

Using the prediction algorithm proposed in Chapter 4, we can predict the link status between two neighboring mobile nodes if the two nodes maintain their movement patterns (speed and direction) for a period of time. Applying the prediction algorithm to the DSR protocol, if a node predicts that the link with its neighbor that it received a packet from will break, it can inform the source node of the packet. Then the source node can find a new route to send other packets to avoid packets being dropped over the

broken link. The following sections describe how the prediction algorithm is implemented in the DSR protocol. Because we use NS2 to simulate ad hoc network, the modification is based on the implementation of DSR in NS2 version 2.1b6 (Linux).

5.1.1 Prediction of Link State

As discussed before, the mobile node only monitors unicast packets that come from its previous hop nodes. In the DSR protocol, the unicast packets are data packets, Route Reply and Route Error messages. Every mobile node maintains a table that contains its active neighboring node addresses, the signal power values of received unicast packets, the reception time of these packets, and the link break prediction time calculated by the prediction algorithm.

Every time a node receives a unicast packet, it puts these parameters (previous hop node address, packet signal power and reception time) in its table and updates the prediction time (obviously, if two nodes maintain their movement patterns, the prediction time will be the same). Only three consecutive packets with decreasing signal power values can be used to compute the prediction time. At the beginning, every prediction time for the link with active neighboring node is set to a default value that is bigger than the simulation time, indicating that the link will be valid all the time. The prediction algorithm uses a “sliding window” with a window size of three. Every time node B receives a packet from node A, it moves the window one forward to compute the prediction time by using the latest three packets. Because link failures only happen when two mobile nodes are moving away, when they are moving closer, the packet’s received signal power value will increase. Then the prediction algorithm will set the “window”

size to one, not compute a prediction time, and set it to a default value. When it receives a packet with a lower signal power strength than the first one's, as discussed in Chapter 4, two packets are not enough for getting an accurate prediction time. In this case, the prediction time will also be set to a default value. We will reset the window size to three again if the following packets' power signal values are getting smaller, and the prediction algorithm calculation will be applied. As shown in Chapter 4, the prediction algorithm needs at least three packets that are sent during the time when two nodes maintain their movement patterns. If during this time, one or both mobile nodes change their movement patterns, (two nodes are still moving away, but may slow down or speed up), the prediction algorithm can yield inaccurate result. According to the mobility model used in NS2, a mobile node can only be in one of two states: pause or move at a constant speed to a fixed destination. When the node receives at most another three packets from its previous hop node after the movement patterns changed, it will obtain a new accurate prediction time again.

5.1.2 Initiating Route Error Messages

Every mobile node maintains a Link Table. Each entry of the Link Table has the source node address; previous hop node address and the packet reception time. Because only the source nodes of active routes will be informed of the bad link A-B, and different routes may share a common link as shown in Figure 5.1, there may be entries that have the same previous hop nodes but with different source nodes in the Link Table. Maintaining the source nodes in the Link Table can prevent node B from repeatedly sending Route Error messages to the same source node in case it received more than one packet from node A

after the link A-B enters critical state.

When mobile node B receives a unicast packet from its previous hop node A, it will check its link status with node A. If the prediction time is equal to or less than the critical time, node B first will check if the source node and previous hop node pair is in its Link Table. If it found one, node B already sent a Route Error message to the source about link A-B status recently, so it does nothing. If it cannot find one, node B will insert the source node and previous hop node pair with the current time into the Link Table. Then node B sends a Route Error message to the source node of this packet. This Route Error message has the same format as the normal DSR Route Error message. The only difference is that the normal Route Error message is sent by the upstream node of the broken link, i.e. node A, the new Route Error message is sent by the downstream node of the soon-to-be-broken link. To differentiate these two kinds of Route Error messages, we call the latter prediction Route Error message. As the DSR protocol specifies, when a Route Error message is sent to the source node, the nodes on its route will remove the routes with link A-B from their route caches.

5.1.3 Handling Route Reply Message

When a source node receives the prediction Route Error message, like in the original DSR protocol, if it cannot find an alternative route that does not contain the bad link A-B, it will initiate a Route Request message and broadcast it. This Route Request message will piggyback the bad link A-B with it. When other mobile nodes in the network receive this message, they will also remove the routes that contain this bad link from their route caches. This mechanism prevents the intermediate nodes from sending a Route Reply

message that includes the bad link from their route caches.

In the original DSR protocol, when a mobile node receives a Route Request, it will send a Route Reply to the source node if it is the destination node of this Route Request or it has a route to the destination in its route cache. When the source node sends a Route Request message, the A-B link is still alive, so there are still some Route Request messages that go through this link, finally some Route Reply messages will have routes that contain the A-B link. When a source node receives this kind of Route Reply messages, it may use this bad link again, and a packet still will be dropped.

One solution to this problem is that every node maintains a bad link table. These bad links are derived from the Route Error messages it received. When a source node receives a Route Reply message, it will check if the new route has any link that is listed in the bad link table. If it is, the node discards this Route Reply message. The drawback of this method is that there is unnecessary traffic produced by the A-B link, and other nodes may snoop the Route Reply message with a bad link. Some time later, these nodes may use this route to salvage packets, then packets will be dropped.

A simpler solution is to use the Link Table in every mobile node. When node B receives a Route Request message, it first checks if the previous hop node is in the link table. If it is, it just stops propagating this Route Request message further and does not reply from its route cache, and then the new route will not contain the A-B link anymore. This method will also reduce some Route Discovery traffic.

Because of the random characteristic of the mobile node mobility, two nodes moving apart for a while, sometime later may move closer. A time-out mechanism is applied to the Link Table entries. Entries will be removed when they time out. If the two nodes are

still moving away, one node cannot be the next hop of the other. If they are moving closer, they may become neighbors in a new route again.

5.1.4 Parameter Setting

Two parameters should be set for the implementation of the prediction algorithm. One is the critical time. According to the simulation result and analysis of Chapter 4, the critical time is set to 1 second. The advantages to choosing a smaller critical time over a bigger one are:

First, closer to the actual link breakage, more accurate prediction results will be obtained and false prediction due to movement pattern changes or lack of packets received can be avoided.

Second, before using a new route to continue sending packets, the current route is still valid. The source node can use it as long as possible.

The other parameter is the time-out value for the Link Table. The time-out mechanism will allow two nodes moving closer, after they are moving apart, to become neighboring nodes in new routes again. Because we already set the critical time to 1 second, the link is supposed to be broken in less than 1 second. Also node B should receive a Route Request message from a source via node A in less than 1 second if the source node sends it, then this message will be blocked by node B. We therefore set the time-out to 2 seconds.

5.1.5 Preliminary Simulation Results

After implementing the prediction algorithm in NS2, a first simulation is done for the DSR protocol. The example we picked is 50 mobile nodes, 20 source nodes, CBR traffic,

64 bytes per packet, 4 packets per second, as before. Scenario settings are simulation area of 1500x300 meters; pause time 0 second, maximum speed is 20 meters per second. The reason to choose such pause time and speed is that in high mobility we can expect more data packets to be dropped, so we can have more cases to study. Simulation time is 900 seconds.

Table 5.1 is a simulation result comparison between using the original DSR protocol and using the DSR protocol implemented with prediction algorithm. Both use the same scenario files (packet generation and node mobility). The main parameters to compare are total number of dropped data packets (total dropped packets), total number of dropped data packets because of no route (total no route dropped packets), total control messages sent (Route Request messages, Route Reply messages, Route Error messages) and hop count ratio. The hop count ratio is the same as defined in Chapter 3. The percentage numbers are all calculated as (original-current)/original. The active link parameter means that the link where a packet was dropped experienced packet transmission in the second prior to the dropped packet, as described in Chapter 3.

Table 5.1 Simulation Result Comparison of Prediction Algorithm

	Original	Current	Percentage %
Total Data Packets Dropped	1315	952	27.60
Total Data Packets Dropped (No Route)	1153	831	27.93
Total Control Messages	26723	32465	-21.49
Hop Count Ratio	1.088	1.086	0.18
Active Link (less than 1 second)	470	75	84.04

From Table 5.1 we can see that both total dropped packets and total no route dropped packets are reduced by around 27%. The total control messages increased by about 21%. The hop count ratio is almost the same. The active link parameter is reduced

significantly.

When a node makes a link prediction and sends a Route Error message to the source node, even with the same scenario files, the routes that packets take may totally differ for the simulation using the original protocol and the simulation using the prediction algorithm. So we cannot compare the control messages on a one-to-one basis. On the other hand, the prediction algorithm will cause more control messages:

- The prediction Route Error message sent by the downstream node of the soon-to-be-broken link, one hop more than the original protocol.
- The false predictions as discussed in Chapter 4, and the extra Route Request and Route Reply it may cause.

The active link parameter shows that the modified DSR protocol uses the prediction algorithm well. The more active links, the more opportunity to obtain the required latest signal power values, and more accurate predictions can be made. The remaining active link number suggests that there are still some packets dropped on some links with prediction. Looking into the simulation trace file, we find out that the main reasons are:

1. Predictions are not accurate because one or two nodes' movement patterns changed as discussed in Chapter 4.

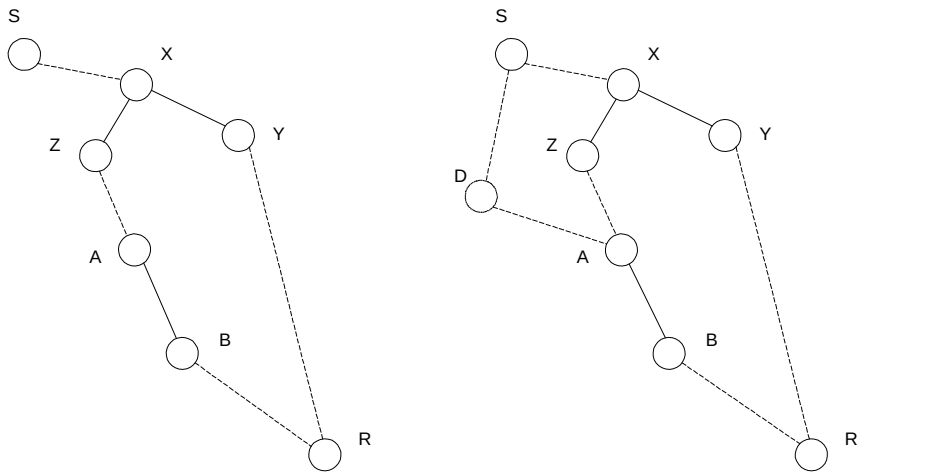


Figure 5.2 Using Bad Link Because of Salvaged Packet

2. The source node of the packet did not receive the prediction Error message because the packet was salvaged.

   The dotted line in Figure 5.2 indicates that there may be more than one hop between two nodes. As illustrated in Figure 5.2 (a), source node S sent a data packet to destination R. X is an intermediate node. The next hop of X in the original route would be node Y. But at this time the X-Y link is broken, and X salvages this packet. Suppose that node Z becomes the next hop of X, and node A and B are intermediate nodes. When X salvaged this packet, it sent a Route Error message to S, and set the first address of the source route of the packet it salvaged to itself. So when node B makes a prediction and sends out a prediction Route Error message, the destination of this message is node X, instead of node S. The source node is not aware that the link A-B will fail. When it received a Route Error message from X, if it had an alternative route that contained the bad link A-B and uses it to send packets, node B will receive a packet from A again. Node B will notice that the node A and S pair is already in the

Link Table, it will not initiate a prediction Route Error again, and finally a packet would be dropped at this link.

3. A node that salvaged the data packet used a route containing a bad link.

   A similar situation is caused by salvaged packet. In Figure 5.2 (b), suppose that node B already sent a prediction Route Error message to source node S via node D, and S received it. Because node S had an alternative route, which does not contain the bad link A-B, to the destination via node X and Y; it uses it to send packets. Sometime later, link X-Y breaks, and node X happens to have a route in its route cache via Z and containing link A-B. Because node S had not sent a Route Request that piggy backed link A-B error before, node X is not aware of this bad link. It will use this route to salvage the packet. At this time the link A-B might have broken, then the packet will be dropped.

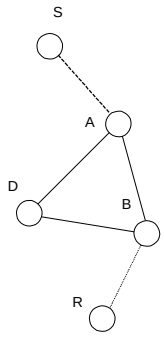


Figure 5.3 Packet Dropped Because of Route Shortening

4. The DSR route shortening mechanism may allow bad links in a new route. The DSR protocol in NS2 implements a route-shortening feature. The basic idea is that there is a route S-B-C-D-E-F-G-R, S is source, R is destination. When node C and node E are moving closer, if finally they are in radio transmission range, node E

will become aware of this and send a message to S to shorten the route. The new route becomes S-B-C-E-F-G-R.

In a route illustrated in Figure 5.3, B sends a prediction Route Error message to source S, if S has no alternative route to R, it will broadcast a Route Request. Because the link A-B is not allowed in the new route according to the implementation, then link A-D and D-B could be in the new route. When the next packet uses this route, if node A and B are still within the radio transmission range, the route shortening mechanism would be triggered and nodes A and B became neighbors again in a new route. Finally a packet will be dropped. The extreme case is when source and destination are one hop away. The source is always aware that the destination is its neighbor.

5. Previous Route Reply and prediction Route Error message arrive at source node in the wrong order.

   This situation is more complex. To explain it clearly, artificial timestamps are used to describe the whole process, because real packet transmissions finish in dozens of milliseconds. Supposed source node S had sent a Route Request before, it got a new route that contains the link A-B. At time 3, node X sends a Route Reply to node S that contains the link A-B to respond to the previous Route Request. At time 3.5, the link A-B enters critical status, node B sends a prediction Route Error message to node S. Node S receives the Route Error message at time 4.0. Because of the network delay, the Route Reply sent by node X arrives at node S at time 4.1. S uses the route of this Route Reply, and then a packet would potentially be dropped.

According to the simulation results in Chapter 4, packet salvages appear frequently in the DSR protocol. If the prediction Route Error messages is sent to the packet sources, instead of the salvage nodes as described in situation 2, more dropped packets can be saved. Based on the first implementation of the prediction algorithm, if the salvaged node received the prediction Route Error message, and the source node still uses the bad link A-B to send data packets, the link A-B might be still valid at that time. Then the downstream node should be allowed to send more than one prediction Route Error messages to the same source node. This will provide more opportunity for the source node of the packet to receive prediction Route Error message, so it will not use the route that contains the bad link, though it cannot entirely avoid situation 2.

To implement this solution, every entry in the Link Table adds a counter. The counter increases by one every time it receives a packet with the same source-previous hop pair and sends a prediction Route Error message if the link is in critical state. The upper bound of the counter is set to three, so the maximum number of prediction Route Error messages a node can send to the same source is three. Experimental result shows that sending this message three times can save more dropped packets than sending it twice and only increases the number of control messages a little. It is not necessary to send even more error messages. According to the parameter settings of the implementation, the link A-B will already be broken if the source node has not received prediction Route Error messages after three attempts.

We still use the same scenario file to show the improvement of the prediction algorithm implementation. Table 5.2 is the comparison of parameters between sending a prediction Route Error message once and sending it at most three times. The other

settings are the same for the implementation.

Table 5.2 Comparison of Sending Prediction Route Error Message

	Original	Once [1]	Three Times[2]	Percentage % [3]
Total Data Packets Dropped	1315	952	827	13.13
Total Data Packets Dropped (No Route)	1153	831	702	15.52
Total Control Messages	26723	32465	33710	-3.83
Hop Count Ratio	1.088	1.086	1.085	0.01
Active Link (less than 1 second)	470	75	41	47.89

1.Sending prediction Route Error message at most once for a source-upstream nodes pair

2.Sending prediction Route Error message at most three times

3.Increase percentage of Three Times to Once

Results in Table 5.2 show that at a small cost of increasing control messages, both total dropped packets counts are reduced significantly, the active links counts is also reduced. Because sending a prediction Route Error message is proactive, a node sends it only when it received packets from the same source-previous hop pair; the total control messages will not increase quickly. Based on the simulation result and analysis, in the rest of the DSR simulation, the prediction Route Error message will be allowed to be sent at most three times for the same source-upstream nodes pair.

Case 5 happens when a previous Route Reply with bad link and a Route Error message arrive at the source node in the wrong order, so the source node uses a route with bad link again. It happens not only in the prediction algorithm implementation, but also in the original DSR protocol. One way to remove this problem is that every source node holds a Source Table. When a source node receives a Route Error message, it puts the two nodes' addresses of the bad link and the reception time of this message into the table. When it receives a Route Reply, it will check if the route in the message has any link in the Source Table. If it has, the source node will discard it. Similar to the Link

Table, when the entry in the Source Table times out, it will be removed in case the two nodes of the bad link move closer late. In DSR, mobile nodes snoop the source route in the data packets and control messages that they or their neighbors transfer; the same procedure will be applied to different snoop processes. In order to verify the idea, at first, the mobile node only checks the Route Reply message when the destination is itself.

Table 5.3 shows a comparison of two examples with or without implementing the Bad Link Tables. Both scenario files use a pause time of 0 second, maximum speed up to 20 meters per second, 50 mobile nodes, 20 source nodes.

Table 5.3 Comparison of Simulation Results with and without Source Table

	Scenario 1	Scenario 2
Total Data Packets Dropped*	906	769
Without Source Table	827	780
Improvement (%)	-9.55	1.41
Total Data Packets Dropped (No Route)*	780	649
Without Source Table	702	662
Improvement (%)	-11.1	1.96
Total Control Messages*	33035	26587
Without Source Table	33718	26263
Improvement (%)	-2.03	1.23
Active Link (less than 1 second)	41	18
Without Source Table	75	21
Improvement (%)	45.33	14.29

* Data from DSR implementing Source Table

From Table 5.3, we do not achieve much gain and even get negative result when implementing the Source Table. But for both examples, the active link number all decreased further. The reasonable explanation of these results is that case 5 happens rarely, and simulation results are subject to fluctuation because of choosing different routes in the network. Based on the simulation results, the Source Table method did not help to reduce dropped packets, and will not be used in the implementation.

5.2 Pro-active Route Maintenance in AODV

First we briefly review route maintenance in the AODV protocol. As in DSR, it is based on the NS2 version 2.1b6 implementation that is a little different from the AODV draft. In AODV, each node keeps track of its active next hops by using available link or network layer mechanisms. When a node detects a link breakage for the next hop of an active route in its routing table, it will bring down this route, and initiate a RERR (called Unsolicited Route Reply in NS2) message with the unreachable destination node address. In this RERR packet, the hop count to the destination is set to infinite; the destination sequence number is increased by one. Then the RERR message is unicast to every active neighboring node of this node that shares the same destination. When a node receives a RERR message, if the source of this message is its next hop, it will bring down the route that has the same destination as shown in the message. Also it initiates a RERR message with this destination address and unicasts it to its active neighbors that share the same destination if there is any. Finally all the precursors of the broken link are informed. The source node will initiate a Route Request if it still has data packets to that destination.

A modification to the original AODV will be made to implement the prediction algorithm. Because our prediction algorithm is independent of the routing protocol, AODV and DSR are both important routing protocols for ad hoc network, and AODV can provide multicast. When a node predicts a link breakage, this node will not bring down its route to this destination because the rest of the route is still valid. Its upstream nodes of this route cannot remove this route entry from their route tables when they are informed of the link status, because at that time this link is still connected. While a node initiates a RERR message and the source node finds a new route, this route could still be

used for transmitting packets.

To represent a new link status, the route flag of each entry in the routing table should add a new state: RTF_PREDICTION. As in DSR, each node maintains a Link Table. Each entry in the Link Table has previous hop node address, destination node address of the packet and packet reception time. The purpose of having a Link Table is to prevent a node from sending multiple RERR messages for the same link-destination pair and replying a route with bad link for the Route Request. Also, there is a time-out for each entry of the Link Table. Then in case two nodes are moving closer some time later, they still can be neighbors in a new route.

When a node receives a unicast packet and knows that the link is in critical state, first it will check if the previous hop node-destination pair is in its Link Table. If it is not, it will add this pair to its Link Table and initiate a RERR messages (called prediction RERR message) and unicast it to its active neighbors that share this destination. This prediction RERR message is almost the same as the original one, except that a flag is used and set to “Prediction_ERROR” to differentiate two kinds of RERR messages.

When a node receives a RERR message, it will know from the flag in the packet that it is a prediction RERR message. If the source of this message is its next hop, it will not remove the route that has the same destination as in the message. It just sets the route flag to RTF_PREDICTION of this entry in its routing table. So this route is still available and valid. Finally the source node of this soon-to-be-broken route will receive the prediction RERR message. It will initiate a Route Request message and broadcast it as the original AODV protocol does. If a node receives a Route Request, it will not reply to it when the route flag of the entry is RTF_PREDICTION, if it happens to have a route to that

destination, because this route will soon break. It will further broadcast it as in the original AODV protocol. When the entry with the same destination in the Link Table times out, this node will also set the hop count to infinite to invalidate this route. The node that initiated the prediction RERR message will handle the Route Request message differently. When it receives a Route Request message, it will check its Link Table to see if the previous hop of this message can be found. If it is found, this Route Request message will be dropped. It will prevent not only this node from replying to the Route Request from its routing table, but also from broadcasting further Route Requests, otherwise the source node may get a route with a soon-to-be-broken link.

In AODV, a prediction RERR message will cause some related unicast prediction RERR messages that precursor nodes may send. To limit this kind of control messages, every node will monitor the AODV unicast control message (Route Reply and RERR), but will not trigger the pro-active route maintenance mechanism. Also, there is no packet salvage mechanism in AODV, sending one prediction RERR message for a link is enough.

Due to the time limit, the prediction algorithm for AODV in NS2 will not be implemented in this thesis. As a good candidate of ad hoc network protocols, AODV implementation is worth to be considered as a future work.

5.3 Experimental Results

After the first set of modifications and experiments, more simulations are run with the DSR protocol with prediction algorithms to see the improvement and overhead caused

under different scenarios. Some example scenarios are picked for the simulations. Because more packets are dropped because of no route in high mobility cases, most of the simulation would be based on the scenarios with a maximum speed up to 20 m/s. As a contrast, low mobility cases (long pause time and lower speed) are also presented. Totally five scenarios have been simulated, each with seven randomly generated scenario files. We would not run all the scenarios discussed in Chapter 3, instead, focus on the scenarios of high mobility cases, which could demonstrate the improvement well. In order to allow for fair comparison, these scenario files are the same as used in Chapter 3. The data traffic file is also the same: CBR, 4 packets per second, 64 bytes per packet. All the simulations run in an area of 1500x300 meter, totally 50 mobile nodes. These five scenario files are:

20-301: 20 sources, pause time: 30 seconds, maximum speed up to 1m/s;

20-30020: 20 sources, pause time: 300 seconds, maximum speed up to 20m/s;

20-3020: 20 sources, pause time: 30 seconds, maximum speed up to 20m/s;

20-020: 20 sources, pause time: 0 second, maximum speed up to 20m/s;

30-020: 30 sources, pause time: 0 second, maximum speed up to 20m/s;

Table 5.4 and Table 5.5 are results of simulations for different scenarios. The parameters in the table are the sum of the values from seven scenario files.

Table 5.4 Simulation Results of DSR Protocol with Prediction Algorithm (1)

	20-301	20-30020	20-3020
Total Data Packets Dropped[1]	658	1800	6507
Original[2]	751	2371	8907
Improvement (%)[3]	12.344	24.208	26.979
95% Confidence Interval	(-10.258, 34.947)	(5.723, 42.694)	(20.776, 33.181)
Total Data Packets Dropped (No Route)	229	1214	5155
Original	416	1947	7490
Improvement (%)	43.863	37.207	31.008
95% Confidence Interval	(27.940, 59.786)	(27.300, 47.114)	(23.217, 38.798)
Total Control Messages	19247	57785	212485
Original	16674	47915	168482
Increase (%)	18.696	20.244	25.359

95% Confidence Interval	(3.909, 33.483)	(10.680, 29.807)	(17.767, 32.951)
Hop Count Ratio[4]	1.061	1.077	1.086
Original	1.062	1.080	1.086
Increase (%)	-0.159	-0.270	-0.026

Table 5.5 Simulation Results of DSR Protocol with Prediction Algorithm (2)

	20-020	30-020
Total Data Packets Dropped	5577	9494
Original	7481	14476
Improvement (%)	24.67	34.295
95% Confidence Interval	(17.500, 31.840)	(30.217, 38.372)
Total Data Packets Dropped (No Route)	4680	8072
Original	6582	12934
Improvement (%)	27.805	37.466
95% Confidence Interval	(20.063, 35.546)	(33.354, 41.579)
Total Control Messages	181547	276265
Original	146714	228191
Increase (%)	23.926	21.720
95% Confidence Interval	(17.187, 30.664)	(17.210, 26.231)
Hop Count Ratio	1.079	1.080
Original	1.081	1.076
Increase (%)	-0.199	0.372

1. Simulation result from DSR protocol implemented with prediction algorithm
2. Simulation result from original DSR protocol
3. Average values of seven simulations improvement
4. Both Hop Count Ratios are average values of seven simulation results

5.4 Discussion of the Simulation Results

From Table 5.4 and Table 5.5, we draw the following conclusions:

- Every simulation result shows improvement in total number of no route dropped packets. The improvement is at least above 20%, and up to 44.95%.
- The control messages increase up to 33.5%.
- The variation of hop count ratio is below 1%, and is considered to be the same.
- The improvement ratio on the total dropped packets is less than total no route dropped packets. There are three cases in the low mobility scenarios (two in 20-301, the other in 20-30020) where total dropped packets increased because of packets dropped in the network interface queue of a few nodes. The two exceptions also result in a negative

final result for 20-301 scenario. Because in lower mobility scenario, the total number of dropped packets in a simulation is only in the dozens, the negative results may be caused by the random fluctuations.

- From the simulation results, we find that for the same mobility file (0 second pause time and up to 20 m/s speed), scenario with more data sources result in more improvement than with fewer data sources. For example, 30-020 case has more improvement than 02-020 case (based on one-to-one simulation result comparison). Because for 30 sources, nodes can have more information of their previous hop nodes to make prediction.
- The total number of dropped packets has been reduced significantly except few cases in low mobility scenarios which original have not many dropped packets. The increased control messages do not cause network congestion and packets loss.

According to the simulation results, the prediction algorithm implemented in DSR routing protocol can significantly reduce the total number of dropped packets due to link breakage by at least 20%, though with a cost of increasing the number of control messages by less than 33.5%. While more consistent improvement ratios are achieved in high mobility scenarios, the simulation results show that both high and low mobility do not suffer any negative consequences.

5.5 TCP Simulation

Since TCP/IP is the standard network protocol stack on the Internet, its use over mobile ad hoc networks is a certainty. TCP adapts to the network changes. The TCP sender assumes that all packet losses are caused by congestion. Thus, when a link on a TCP

route breaks, the TCP sender reacts as if congestion was the cause, reducing its congestion window and, in the instance of a timeout, backing-off its retransmission timeout. Therefore, route change due to node mobility can have a determinable impact on TCP performance.

According to our experimental results with CBR traffic, the DSR protocol implemented with the prediction algorithm can significantly reduce the number of dropped packets by predicting the link availability. A simulation has been done to see how much throughput improvement can be achieved for TCP when using the DSR protocol implemented with the prediction algorithm.

NS2 can generate random TCP traffic connections between mobile nodes. The traffic pattern we used is 50 mobile nodes, 20 sources, packet size 64 bytes. NS2 sets the window size to 32. The application is FTP that is built on TCP. We use the same node mobility scenario file as in the CBR simulation with pause time 0 second and speed up to 20 m/s. Because the TCP simulation will generate a much bigger trace file and needs longer execution time than CBR, the simulation time is set to 500 seconds. A Perl script is written to extract the total packets received from the simulation trace file.

Table 5.6 shows the TCP simulation results. At first, the parameters used are critical time 1 seconds and sending prediction Route Error message at most three times for the same source-previous hop pair as used in the simulations with CBR traffic. The results show that the total number of received packets using the prediction algorithm decreased compared to the original protocol.

When we analyzed the simulation trace file, we found some differences in the simulation results between CBR and TCP traffic. First, a TCP sender does not send data

packets at constant rate as in CBR. So for the node that initiates a prediction Route Error messages, it is not necessary to send multiple ones, because the destination of these messages may be the same node that salvaged multiple packets. Second, the traffic is high in TCP, so the delay for packet delivery is much higher in TCP, even though the routing messages have high priority. The trace file shows that the prediction Route Error message did not arrive at the source node before the next data packet is sent, which resulted in duplicated Route Error messages and more traffic.

We changed the parameter settings to adapt to the TCP traffic characteristics. The critical time is set to 2 seconds so that there is enough time for the source to receive the prediction Route Error message. The maximum number of Route Error message is reduced to 2 to reduce traffic. The simulation result that is listed in Table 5.6 (2,2) column shows some improvement compared with original protocol.

Table 5.6 TCP Simulation Results

(Critical Time, Max Error Message No[1].)	(1,3) original[2]	(1,3)	(2,2)
Total Data Packets Received	197911	196202	209040

1. Maximum number of Route Error message can be sent for the same source-previous hop pair

2. Result using original DSR

One conclusion that could be obtained from our first set of simulations is that in DSR protocol, the improvement based on maintaining a valid route by using our prediction algorithm will not be as significant as with CBR traffic. The prediction algorithm reduced the link breakage possibility on active routes as shown in CBR simulations. But as discussed in Chapter 4, active links only account for about half, or even one third in high mobility, of the broken links; there are still many broken links we cannot predict. The main problem may be caused by DSR's aggressive route cache strategy as suggested in simulation results [6]. The published results show that for a single TCP connection

turning off replying from caches will improve the TCP throughput performance. Furthermore, the TCP time-out mechanism combined with the stale route in the cache will reduce the TCP performance. Figure 5.4 illustrates a scenario [30]. First a route is broken; the source node gets the route from its route cache or from a nearby node. After a time-out, the TCP sender sends a packet on the new route. However, this route is also broken after it was cached. Then TCP will use another time-out; DSR will perform another route discovery. The process will repeat until a valid route is found.

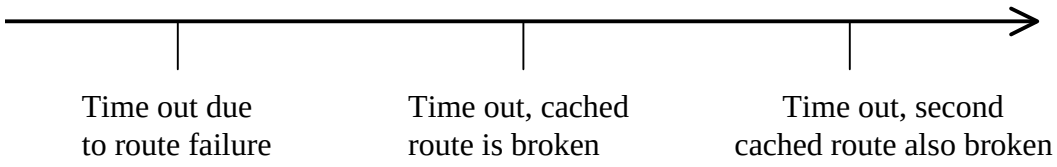


Figure 5.4 Stale Routes Degrade TCP Performance

Our simulation results show that the stale routes degrade TCP performance. There is one source-destination pair that never had a single successful transmission because of route failure. The source node tried to send packets several times and all failed because of link breakage over intermediate links. The TCP performance may be improved much if the prediction algorithm can be used to remove the stale routes in route caches. We leave it as future work.

Chapter 6 Conclusion and Future Work

In this thesis, a link status prediction algorithm is developed and implemented in the DSR ad hoc network routing protocol. Every mobile node monitors received unicast packets' signal power strength. It will predict the link status after receiving three packets if the two nodes of this link maintain their movement patterns during this time. When the link enters critical state, the node will send a prediction Route Error message to the source node of the packet. The upstream nodes of this route will remove routes with this bad link. To increase the possibility that the source node of the original packet receives the message, instead of the node that salvaged the packet, the Route Error message will be sent more than once if there are still packets coming from the same previous hop-source node pair. After receiving a Route Error message, the source node will initiate a Route Request message and broadcast it if it cannot find an alternative route to the destination and also has packets to deliver. The link is also piggybacked with this Route Request message; so that the mobile nodes receiving this message will disable all the routes with the bad link. When a node receives a Route Request, it will check if it comes from an upstream node of a bad link. If it is, the Route Request is discarded.

Simulations have been done for low and high mobility scenarios. The results show that the total no route packets dropped are reduced by at least 20%, and up to 44.95%. At the same time, the total control messages increased up to 33.5%, and the hop count ratio stayed at the same level. So the prediction algorithm significantly reduces the packet drop rate at a cost of increasing routing messages overhead.

Besides the prediction algorithm in AODV, future work may focus on saving more

dropped packets by increasing the prediction ability and on decreasing total control messages. Here are some basic ideas:

- In this thesis, a mobile node only monitors unicast packets over one hop of a route. So if two nodes did not communicate before or the movement pattern changed since their last communication, the prediction cannot be made because not enough information is available. We learn from the simulation results in Chapter 4 that in high mobility scenarios, in the DSR protocol, more than half of the no route dropped packets happen when these links are not active links. To improve prediction ability and provide more link status information, the multicast packets like Route Request messages can be monitored. Because the wireless network interface card can be set to the promiscuous mode, all the packets, even MAC layer packets in a mobile node receiving range also can be monitored. Then a node can have more link status information with its neighboring nodes, even if they do not communicate directly. But in the DSR protocol, the route maintenance only works for active links. A node cannot initiate a prediction Route Error message for an inactive link; otherwise many unnecessary control messages will be produced. This kind of prediction Route Error messages for inactive routes can be piggybacked in Route Request messages. For example, node A and node B are neighboring nodes, and are used as a part of a route for a while. The link was not in a critical state. Then, the route switched, link A-B was not active. Suppose that node A and node B are moving apart, node B is monitoring the packets node A sends. When link A-B enters into the critical state, if there is some Route Request message relayed by node A, in that period of time, this bad link A-B will be piggy-backed with this message. Then more nodes will be

informed of this bad link and can remove the routes with this link from their route caches. Also this bad link can be piggybacked in unicast control messages like Route Error and Route Reply, because nodes snoop the packets neighboring nodes transmit as the original DSR protocol proposed, so some nodes will be informed. The DSR protocol aggressively uses route caches and lacks mechanism to expire stale routes in the cache. It may help at low mobility rate, but will reduce the throughput performance in more "stressful" situations [11], because stale routes from the route cache used for Route Reply or to salvage packets will cause more packets to be dropped. The above mechanism does not increase the number of control messages (but will increase control message's size), and updates route caches to reduce stale routes, finally reducing the possibility that routes with bad links are used, though this possibility can not be reduced to zero. We believe it will reduce the number of dropped packets significantly at low cost.

- Reducing control messages may reduce dropped packets (because of full queues) and increase throughput. Closer to the link breakage time, more accurate predictions can be made, because there is less opportunity for two nodes to change their movement pattern. So false predictions and the Route Request/Route Reply messages can be avoided. Also if the time when a node sends prediction Route Error is closer to the link breakage time, the less possibility there is for the scenarios discussed in Section 5.1.5, so more dropped packets can be saved. More experiments could be done to find a better critical time (in this thesis, the critical time is set to 1 second) though it may be different for high and low mobility scenarios, different number of nodes. In addition, a node does not have to send a prediction Route Error message when it

receives a Route Reply message. This will reduce the number of control messages.

- More experiments can be done on the effect of different number of mobile nodes, data sending rate, and simulation area on the reduction of total number of dropped data packets.
- In this thesis, once a source node receives a prediction Route Error message, it will stop using the soon-to-be-broken route though this route is still valid. If it cannot find an alternative route, it will go through the Route Request, Route Reply procedure. During this time, data packets have to be in its send buffer, which will reduce network throughput. Future work can be done to continue using this old route until a new route is found, a process like “soft handover”. A possible solution is when the source node sends a Route Request message with bad link piggybacked; it still sends data packets via the old route. When nodes send a Route Reply to the source node, the bad link copied from the Route Request message will be piggybacked with this message. Then the source node finally removes the routes with the bad link, and uses the new route to send packets when it received the Route Reply. This Route Reply message also helps neighboring nodes along the new route to remove a bad link if they happen to learn that route from the packets.
- Finally, future work could be further improvement on TCP performance. First, to avoid sending duplicate Route Error messages to the same node, a node can compare the destination address of this Route Error message with the previous one with the same source-destination pair, and only sends a Route Error message when they are different. Second, the prediction algorithm can be used to remove stale routes in route caches as discussed before. Then after time-out, the source node still can use a “fresh”

route to send packets. Third, in our simulation scenarios, TCP will generate more traffic than CBR, it is important to find an appropriate critical time in order to inform the source node before the route breaks.

References

[1] Joseph Macker and Scott Corson, IETF Mobile Ad Hoc Networks (MANET) Charter, http://www.ietf.org/html.charters/manet-charter.html.

[2] Charles E. Perkins, IP mobility support, RFC 2002. October 1996. ftp://ftp.isi.edu/in-notes/rfc2002.txt.

[3] Charles E. Perkins, Pravin Bhagwat, Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. Proceedings of the Conference on Communications Architectures, Protocols and Applications, pages 234-244, London, England, August 1994.

[4] Charles E. Perkins and Elizabeth M. Royer, Ad-hoc On-Demand Distance Vector Routing. Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, pages 90-100, New Orleans, LA, February 1999.

[5] David B. Johnson and David A. Maltz, Dynamic Source Routing in Ad Hoc Wireless Networks. In Mobile Computing, edited by Tomas Imielinski and Hank Korth, Chapter 5, pages 153-181, Kluwer Academic Publishers, ISBN: 0792396979, 1996.

[6] Gavin Holland and Nitin Vaidya, Analysis of TCP Performance Over Mobile Ad Hoc Networks. Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, pages 219-230, Seattle, Washington, August 1999.

[7] C.-K Toh. Associativity-Based Routing for Ad-Hoc Networks. Wireless Personal Communications Journal, Special Issue on Mobile Networking and Computing Systems, pages 103-139, vol. 4, no. 2, March 1997.

[8] He Dajing, Jiang Shengming and Rao Jianqiang, A Link Availability Prediction

Model for Wireless Ad Hoc Networks. Proceedings of the International Workshop on Wireless Networks and Mobile Computing, D7-D11, Taipei, Taiwan, April 2000.

[9] William Su and Mario Gerla, IpV6 Flow Handoff in Ad-Hoc Wireless Networks Using Mobility Prediction. Proceedings of IEEE Global Communications Conference, pages 271-275, Rio de Janeiro, Brazil, December 1999.

[10] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu and Jorjeta Jetcheva, A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking, pages 85-97, Dallas, TX, October 1998.

[11] Samir R. Das, Charles E. Perkins and Elizabeth M. Royer, Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks. Proceedings of the IEEE Conference on Computer Communications, pages 3-12, Tel Aviv, Israel, March 2000.

[12] Elizabeth M. Royer and C.-K. Toh, A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks. IEEE Personal Communications Magazine, pages 46-55, April 1999.

[13] L.R. Ford Jr. and D.R. Fulkerson, Flows in Networks. Princeton University Press, Princeton NJ, ASIN: 0691079625, 1962.

[14] Ching-Chuan Chiang and Mario Gerla, Routing in Clustered Multihop, Mobiles Wireless Networks. Proceedings of IEEE Singapore International Conference on Networks, pages 197-211, Singapore, April 1997.

[15] Vincent D. Park, M. Scott Corson, A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. Proceedings of the IEEE Conference on Computer Communications, pages 1405-1413, Kobe, Japan, April 1997.

[16] Vincent D. Park and M. Scott Corson, A Performance Comparison of the Temporally-Ordered Routing Algorithm and Ideal Link-State Routing. Proceedings of IEEE Symposium on Computers and Communication '98, pages 592-598, Athens, Greece, June 1998.

[17] M. Scott Corson and Vincent D. Park, Internet MANET Encapsulation Protocol (IMEP). Internet draft, draft-ietf-manet-imep-spec-01.txt, Work in progress.

[18] John Jubin and Janet D. Tornow, The DARPA Packet Radio Network Protocols. Proceedings of IEEE, 75(1), pages 21-32, January 1987.

[19] David A. Maltz, Josh Broch, Jorjeta Jetcheva and David B. Johnson, The Effects of On-Demand Behavior in Routing Protocols for Multihop Wireless Ad Hoc Networks. IEEE Journal on Selected Areas in Communications Special Issue on Mobile and Wireless Networks, pages 1439-1453, August 1999.

[20] Rohit Dube, Cynthia D. Rais, Kuang-Yeh Wang and Satish K. Tripathi, Signal Stability-Based Adaptive Routing (SSA) for Ad Hoc Networks. IEEE Personal Communications, pages 36-45, February 1997.

[21] A. Bruce McDonald and Taieb Znati, Predicting Node Proximity in Ad-Hoc Networks: A Least Overhead Adaptive Model for Selecting Stable Routes. IEEE Mobile Ad Hoc Computing and Networking, Boston, Massachusetts, August 2000.

[22] Sergio Marti, T. J. Giuli, Kevin Lai and Mary Baker, Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking, Boston, Massachusetts, August 2000.

[23] http://www.opnet.com.

[24] Kevin Fall and Kannan Varahan, editors. NS Notes and Documentation. The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, November 1997.

[25] http://www.isi.edu/nsnam/ns/.

[26] Bruce Tuch, Development of WaveLAN, an ISM Band Wireless LAN. AT&T Technical Journal, 72(4), pages 27-33, July/August 1993.

[27] Daehyoung Hong and Stephen Rappaport, Traffic Models and Performance Analysis for Cellular Mobile Radio Telephone Systems with Prioritized and Non-Prioritized Handoff Procedures. IEEE Transactions on Vehicular Technology, Vol. VT-35, No.3, pages 77-92, August 1986.

[28] B. Narendran, P. Agrawal and D. K. Anvekar, Minimizing Cellular Handover Failures without Channel Utilization Loss. Proceedings of IEEE Global Communications Conference, pages 1679-1685, vol. 3, December 1994.

[29] NS2 manual, http://www.isi.edu/nsnam/ns/doc/index.html.

[30] Nitin H. Vaidya, Tutorial, http://www.cs.tamu.edu/faculty/vaidya/seminars/tcp-tutorial-aug99.ppt.

Appendix

1. The Prediction Algorithm Implementation in DSR Using Network Simulator (NS2)

Configuration

The following configuration was used for performing all the simulations described in this thesis:

Computer: Pentium II, Dell Optiplex Gxa, 128MB RAM

Operating System: Linux Red Hat 6.2

NS2 version: ns-allinone-2.1b6

URL: http://www.isi.edu/nsnam/ns/index.html

To install NS2 simply type "./install" in the directory of extraction folder: ns-allinone-2.1b6.

2. Network Components in a Mobile Node in NS2 [29]

The network stack for a mobile node consists of a link layer (LL), an ARP module connected to LL, an interface priority queue (Ifq), a mac layer (MAC), a network interface (netIF), all connected to the channel. These network components are created and combined in Otcl. Figure A-1 is schematic of a DSR mobile node.

Link Layer: the link layer for a mobile node has an ARP module connected to it which resolves all IP to hardware (Mac) address conversions. Normally for all outgoing (into the channel) packets, the packets are handed down to the LL by the routing agent. The LL hands down packets to the interface queue. For all incoming packets (out of the channel)

the mac layer hands up packets to the LL which is then handed off to node_entry_point. The class LL is implemented in: ns-allinone-2.1b6/ns-2.1b6/ll.{cc,h}

ARP: The Address Resolution Protocol module receives queries from Link layer. If ARP has the hardware address for the destination, it writes it into the mac header of the packet. Otherwise it broadcasts an ARP query, and caches the packet temporarily. For each unknown destination hardware address, there is a buffer for a single packet. In case additional packets to the same destination are sent to ARP, the earlier buffered packet is dropped. Once the hardware address of the packet's next hop is known, the packet is inserted into the interface queue. The class ARPTable is implemented in ns-allinone-2.1b6/ns-2.1b6//arp.{cc,h}

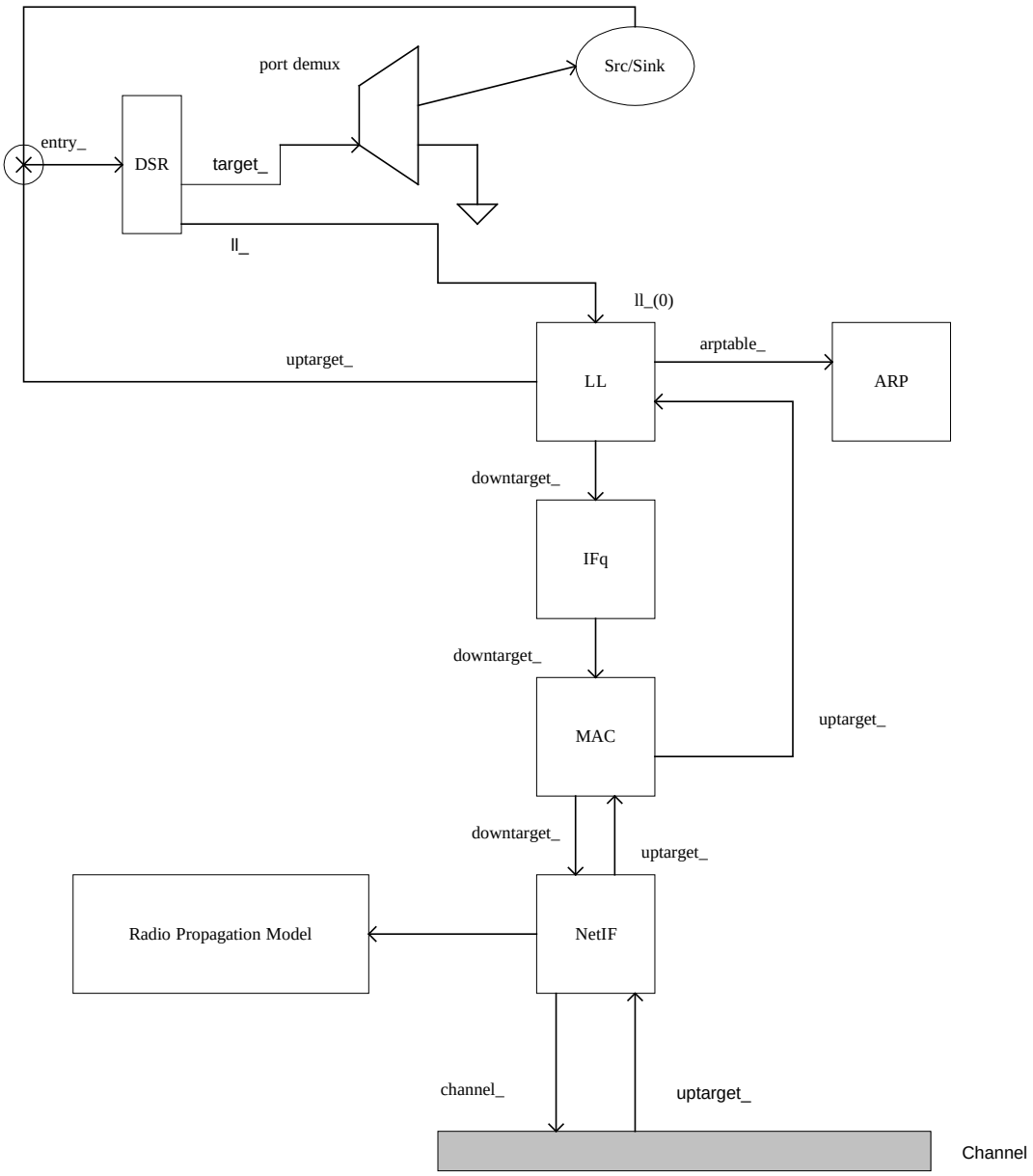


Figure A-1 Schematic of DSR Mobile Node

Interface Queue: The class PriQueue is implemented as a priority queue which gives priority to routing protocol packets, inserting them at the head of the queue. It supports running a filter over all packets in the queue and removes those with a specified destination address. The class is implemented in ns-allinone-2.1b6/ns-

2.1b6/priqueue.{cc.h}

Mac layer: The IEEE 802.11 distributed coordination function (DCF) MAC protocol has been implemented by CMU. It uses the RTS/CTS/DATA/ACK pattern for all unicast packets and simply sends out DATA for all broadcast packets. The implementation uses both physical and virtual carrier sense. The class Mac802_11 is implemented in ns-allinone-2.1b6/ns-2.1b6/mac-802_11.{cc,h}

Network Interfaces: The Network Interface layer serves as a hardware interface that is used by mobile node interfaces to the channel. The wireless shared media interface is implemented as class Phy/WirelessPhy. This interface is subject to collisions and the radio propagation model receives packets transmitted by other node interfaces to the channel. The interface stamps each transmitted packet with the meta-data related to the transmitting interface like the transmission power, wavelength. This meta-data in the packet header is used by the propagation model on the receiving network interface to determine if the packet has minimum power to be received and/or captured and/or detected (carrier sense) by the receiving node. The model approximates the DSSS radio interface (Lucent WaveLen direct-sequence spread-spectrum). The class is implemented in ns-allinone-2.1b6/ns-2.1b6/wireless-phy.{cc,h}

Radio Propagation Model: The Radio Propagation Model uses Friss-space attenuation ($1/r^2$) at near distances and an approximation to Two Ray Ground ($1/r^4$) at far distance. The approximation assumes specula reflection off a flat ground plane. The class is implemented in ns-allinone-2.1b6/ns-2.1b6/tworayground.{cc,h}

Antenna: An omni-directional antenna with unity gain is used by mobile nodes. The class in implemented in ns-allinone-2.1b6/ns-2.1b6/antenna.{cc,h}

Wireless Channel: The wireless channel duplicates packets to all mobile nodes attached to the channel except the source itself. It is the receiver's responsibility to detect if it can receive the packet.

3. The Prediction Algorithm Implementation

Location: ns-allinone-2.1b6/ns-2.1b6/

Files: node.h, node.cc, wireless-phy.cc

The following is a brief summary of the implementation:

- In wireless-phy.cc, when a node receives a unicast packets, and it is the next hop of the packet, it then adds (previous hop address, receive time, signal power strength) to the node's moving away neighbors' table.
- In node.h, the link breakage prediction equation is implemented. Also, it selects receive signals to ensure that only signals with decreasing power strengths can be put into the table.
- In node.h, we added a moving away neighbors' table. Each entry of the table is an instance of the SignalPower data structure which stores the neighbor's address and prediction time calculated from the receive signal power strength.
- In node.cc, initialization table, methods such as retrieve prediction time and store signal power parameter are implemented.
- In wirless-phy.cc, we added two items to the trace file. Once two neighboring nodes in a route are moving away, the previous hop node address and link breakage prediction time will be shown on each receiving node's trace item.

4. DSR Modification

Location: ns-allinone-2.1b6/ns-2.1b6/dsr/

Files: dsragent.h, dsragent.cc

The DSR routing agent inherits from the standard Agent class. The following is a brief summary of the changes that have been made:

- In dsragent.h, set critical time, timer for Link Table entry.
- In dsragent.h, add Link Table of which entry holds source node-previous hop node pair.
- In dsragent.cc, initialize Link Table.
- In dsragent.cc, monitor unicast packets: data packets, Route Reply packets that are intend for itself. Check prediction time and decides whether initialize prediction Route Error message.
- In dsragent.cc, initialize and send prediction Route Error messages. Set the maximum number of prediction Route Error messages can be sent for the same source-previous node pair.
- In dsragent.cc, handle the Route Request based on whether this message is coming from a soon-to-be-broken link.
- In dsragent.cc, time-out for Link Table entry.

5. Creating Mobile Node Movement Scenario Files

The node-movement generator is available under:

ns-allinone-2.1b6/ns-2.1b6/indep-utils/cmu-scen-gen/setdest directory.

Run setdest with arguments as shown below:

./setdest [-n num_of_nodes] [-p pausetime] [-s maxspeed] [-t simtime] [-x maxx] [-y maxy] > [outdir/movement-file]

6. Creating Random Traffic Pattern for Wireless Scenarios

Random traffic connections of TCP and CBR can be setup between mobile nodes using a traffic-scenario generator script. This traffic generator script is available under:

ns-allinone-2.1b6/ns-2.1b6/indep-utils/cmu-scen-gen

It is called cbrgen.tcl. The command line looks as the follows:

ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed seed] [-mc connections] [-rate rate]>[outdir/movement-file]

The rate parameter is only used for CBR packets. Its inverse value is used to compute the interval time between the CBR packets.

To set the data packet size, modify the parameter item in cbrgen.tcl:

"set opt(pktsize) 64"