

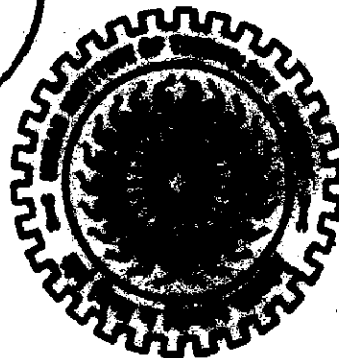
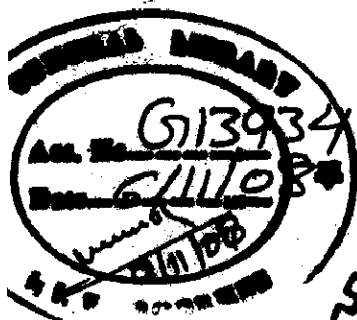
ADAPTIVE MULTIPATH SOURCE ROUTING PROTOCOL IN MOBILE AD HOC NETWORKS

A DISSERTATION

*Submitted in partial fulfillment of the
requirements for the award of the degree
of*
MASTER OF TECHNOLOGY
in
INFORMATION TECHNOLOGY

By

RAMBABU YERAJANA



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

JUNE 2008

Candidate's Declaration

I declare that the work being presented in the dissertation report titled "An Multipath Source Routing Protocol in Mobile Ad hoc Networks" in fulfillment of the requirement for the award of the degree of Master of Science in Information Technology, submitted in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, is an authentic and my own work carried out under the guidance of Dr. A. K. Sarje, Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology

Roorkee, submitted the matter embodied in this dissertation report for the award of the degree.


16/2008
Roorkee.


(Rambabu Yerajana)

Certificate

I certify that above statements made by the candidate are correct to the best of my knowledge and belief.

Roorkee.


21/6/08
Dr. A. K. Sarje,
Professor,
Department of Electronics and
Computer Engineering, IIT Roorkee,
Roorkee -247667 (India).

ACKNOWLEDGEMENTS

I am thankful to Indian Institute of Technology Roorkee for giving me this opportunity. It is my privilege to express thanks and my profound gratitude to my supervisor Prof. A. K. Sarje for his invaluable guidance and constant encouragement throughout the dissertation. I was able to complete this dissertation in this time due to constant motivation and support obtained from Prof. A. K. Sarje.

I am also grateful to the staff of software laboratory and Research Scholar's laboratory for their kind cooperation extended by them in the execution of this dissertation. I am also thankful to all my friends who helped me directly and indirectly in completing this dissertation.

Most importantly, I would like to extend my deepest appreciation to my family for their love, encouragement and moral support. Finally I thank God for being kind to me and driving me through this journey.

(Rambabu Yerajana)

ABSTRACT

Mobile ad hoc networking is rapidly gaining popularity due to the proliferation of miniature yet powerful mobile computing devices. Mobile ad hoc networks do not require any form of fixed infrastructure for hosts to be able to communicate with one another. A source node that needs to communicate with a destination node uses either a direct link or a multihop route to reach the latter. This requires that all nodes must have some basic routing capability to ensure that packets are delivered to their respective destinations. Since nodes may move anytime, then the topology of the network may also change anytime. A major challenge in mobile ad hoc networking is how to maximize data packet delivery in the face of rapidly changing network topology without incurring a large routing overhead.

In this dissertation, a routing protocol for mobile ad hoc networks has been proposed. The proposed protocol is based on Dynamic Source Routing Protocol. Mobile Ad hoc networks with nodes having different processing powers and thus can perform extensive computations apart from forwarding packets for other nodes. These nodes will also have various degrees of battery powers as well. Due to the heterogeneity of the systems in terms of processing and battery powers, naturally, there will be load imbalance. If the workload is distributed among the nodes in the system based on the resources of individual nodes, the average execution time can be minimized and the lifetime of the nodes can be maximized. Our proposed protocol uses congestion status and distributes the load according to the congestion status of the path. Simulation results shows that our proposed routing protocol gives better performance than DSR and AODV in terms of end-end-delay and throughputs

Contents

Candidate's Declaration	i
Acknowledgements	ii
Abstract	iii
CHAPTER 1: INTRODUCTION	1
1.1 Background	1
1.2 Mobile Ad hoc networks overview	2
1.3 Motivation	4
1.4 Statement of the problem	4
1.5 Organization of the report	4
CHAPTER 2: Routing in mobile ad hoc networks	5
2.1 Routing Protocols for Wired Networks	5
2.2 Routing in MANET	6
2.3 Types of Routing protocols	8
2.3.1 Flooding	9
2.3.2 Proactive routing protocols	10
2.3.3 On-demand routing protocols	11
2.4 Multipath routing in MANET	12
CHAPTER 3: Dynamic Source Routing Protocol	13
3.1 DSR Protocol	13
3.2 Route Discovery	13
3.3 Route Maintenance	16
3.4 Route caching	16
3.5 DSR Advantages and Disadvantages	17
3.6 Additional Route Discovery features	18
3.7 Additional Route maintenance features	18

CHAPTER 4: Load Balancing in MANET	19
4.1 Singlepath verses Multipath routing	20
4.2 Multipath Source Routing in Wireless Ad Hoc Networks	21
4.2.1 Loop-free problem	
4.2.2 Packet forwarding and load balancing	
4.3 An Adaptive Multipath Source Routing Protocol	23
4.4 Adaptive multipath source routing protocol: advantages and disadvantages	27
CHAPTER 5: SIMULATION	28
5.1 GLOMOSIM Overview	28
5.2 Use of GloMoSim Simulator	28
5.3 Basic Structure of the source directory	28
5.4 The Application Configuration File	29
5.5 Config.in File	32
CHAPTER 6: RESULTS AND DISCUSSION	35
6.1 Performance Metrics	35
CHAPTER 7: CONCLUSION AND FUTURE WORK	38
REFERENCES	39
APPENDIX: Source code Listing	

1 Introduction

1.1 Background

In the wireless communication systems, there will be a need for the rapid deployment of independent mobile users. Significant examples include establishing survivable, efficient, dynamic communication for emergency/rescue operations, disaster relief efforts, and military networks [2], [3]. Such network scenarios cannot rely on centralized and organized connectivity, and can be conceived as applications of **Mobile Ad Hoc Networks (MANET)**. A MANET is an autonomous collection of mobile users that communicate over relatively bandwidth constrained wireless links. Since the nodes are mobile, the network topology may change rapidly and unpredictably over time. The network is decentralized, where all network activity including discovering the topology and delivering messages must be executed by the nodes themselves, i.e., routing functionality will be incorporated into mobile nodes.

The set of applications for MANETs is diverse, ranging from small, static networks that are constrained by power sources, to large-scale, mobile, highly dynamic networks [2]. The design of network protocols for these networks is a complex issue. Regardless of the application, MANETs need efficient distributed algorithms to determine network organization, link scheduling, and routing. However, determining viable routing paths and delivering messages in a decentralized environment where network topology fluctuates is not a well-defined problem. While the shortest path (based on a given cost function) from a source to a destination in a static network is usually the optimal route, this idea is not easily extended to MANETs.

In wireless computer networks, **ad-hoc** mode is a method for wireless devices to directly communicate with each other. Operating in ad-hoc mode allows all wireless devices within range of each other to discover and communicate in peer-to-peer fashion without involving central access points (including those built into broadband wireless routers).

Mobile Ad Hoc Networks overview

wireless networking, there are two main architectures: infrastructure networks and mobile ad hoc networks. An infrastructure network has a fixed and wired backbone network, it includes cellular networks and wireless local area networks and users are connected via base stations/access points and backbone networks. And second type of architecture, called ad-hoc, does not rely on any solitary infrastructure.

Mobile Ad Hoc Networks (MANET) is a collection of mobile nodes that can communicate with each other using multi hop wireless links without utilizing fixed based-station infrastructure and centralized management [2], [3], and [4]. Each mobile node in the network acts as both a host generating flows or a destination of flows and a router forwarding flows directed to other nodes. In this case a network is formed dynamically through the cooperation of an arbitrary set of independent nodes. There is no prearrangement regarding the specific role each node should assume. Instead, each node makes its decision independently, based on the network situation, without using a preexisting network infrastructure. Ad hoc wireless networks are self-creating, self-organizing, and self-administering.

Mobile ad-hoc networks have certain advantages over the traditional communication networks which described in [2], [3]. Use of ad-hoc networks increase mobility and flexibility, as ad-hoc networks can be brought up and taken down in a very short time. Ad-hoc networks can be more robust than conventional wireless networks because of their non-hierarchical distributed control and management mechanisms.

Mobile ad hoc networks can be categorized depending on their coverage area into four types, Body Area Networks, Personal Area Networks, Wireless Local Area Networks and Wide Area Networks [2].

Figure 1.1 shows an example of mobile ad hoc network and its communication topology. Here circles denote the coverage area or transmission range of responding mobile nodes.

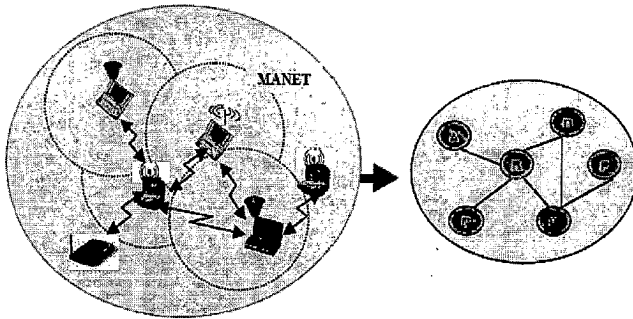


Figure 1.1

Ad hoc networks have less transmission range. Communication paths between nodes which are not in transmission range or radio range of each other are established by intermediate nodes acting as relays to forward data packets toward the destination. In such an environment, it may be necessary for one mobile host to seek the aid of others, in forwarding a packet, due to the limited propagation range.

In ad hoc wireless networks, a remote mobile node interconnection is achieved by a peer level multihopping technique. This implies that the interconnection topology can change dynamically, giving rise to many challenging research issues. In this environment, ad hoc routing is critical and has to be supported before any applications can be deployed for ad hoc mobile networks.

3 Motivation

Multipath routing protocols for Congestion control and load balancing in Ad hoc networks have been a very active research area and many routing protocols have been already proposed. Most of the protocols are concentrated on single path routing. Congestion is the main reason for packet delay and/or packet loss in ANET. Multipath routing protocols give better performance than single path routing protocols in terms of end-to-end delay and throughput.

Our proposed protocol is based on multipath source routing protocol which is based on congestion status of the path.

4 Statement of the problem

The problem is, design and implementation of a new routing protocol in mobile ad hoc networks for load balancing. The aim of this dissertation is to,

Study the various on-demand routing protocol as well as multipath routing protocols in mobile ad hoc networks.

Propose a new routing protocol for load balancing and congestion control in mobile Ad hoc Networks.

Evaluate the performance of the proposed protocol, simulate it and compare the results with existing protocols.

5 Thesis Organization

The rest of the thesis is organized as follows,

Chapter 2, describes the need and difficulties of routing in ad hoc networks and types of routing protocols available.

Chapter 3 gives detailed description of an on-demand routing protocol called dynamic source routing protocol.

Chapter 4, we introduce the concept of load balancing in ad hoc networks.

Chapter 5, we give the detail description of simulator and

Chapter 6, presents simulation results and discussion finally we conclude.

Routing In Mobile Ad Hoc Networks

A Brief Review of Traditional Routing Protocols for Wired Networks

Conventional routing protocols are based on routing tables, which store paths to possible destinations [1], [2]. To guarantee that routing tables are up to date and reflect the actual network topology, nodes continuously exchange route updates and recalculate the paths. Such protocols are divided into two complementary classes:

Distance Vector (DV) algorithms, in which a node sends to its neighbors the whole routing table (its distance vector).

Link State (LS) algorithms [2], [3], in which a node sends to all the other nodes the state of the link with the current neighbors via a reliable flooding.

Route updates are sent periodically or when a topological change is detected. In LS algorithms, each node stores the whole network topology and calculates the shortest path autonomously from the others. Information about the topology is advertised in the form of link-state. The link state indicates whenever the links to the neighbors are up or down and the cost associated with it. Due to flooding, each node eventually receives all the link-states from all the other nodes of the network.

MANET technology, because it provides dynamic and mobile wireless network support, is naturally being considered for use in embedded devices, including human wearable and portable devices. Devices such as compact laptop computers, personal digital assistants (PDAs), and embedded computing systems have been demonstrated running varieties of MANET routing technology. This technology achievement enables the ad hoc formation and maintenance of dynamic wireless infrastructures even among small, embedded devices. However, there are a number of issues related to the use of small, portable devices that deserve consideration when developing and selecting appropriate MANET technology or modes of operation including: _ More limited energy (e.g., possibly battery-/solar-operated devices) _ More limited computing power _ More limited memory _ Increased interference and dynamics (near field RF effects). Limited energy can be a critical operational

ration. For instance, if a battery powered device is naturally power (e.g., into and out of a dormant mode) this may need to be considered as part in the design assumptions of a routing protocol.

Routing in MANET

It is a peer-to-peer wireless network that transmits from computer to computer without the use of a central base station (access point) [1], [2], [3]. Data from one node to another on such a network typically uses an "on-demand routing protocol," such as DSR (Dynamic Source Routing) or AODV (Ad hoc On-demand Distance Vector routing), which generates routing information only when a station initiates a transmission.

Following are typical design goals for ad hoc network routing protocols [1], [3]:

Minimal control overhead: Control messaging consumes bandwidth, processing resources, and battery power to both transmit and receive a message. Since bandwidth is at a premium, routing protocols should not send more than the minimum number of control messages they need for operation, and be designed so that this number is relatively small. While transmitting is twice as power consuming as receiving, both operations are still power drains for the mobile devices. Hence, reducing control messaging also helps conserve battery power.

Minimal processing overhead: Algorithms that are computationally complex require significant processing cycles in devices. Because the processing cycles on the mobile device to use resources, more battery power is consumed. Algorithms that are lightweight and require a minimum of processing from the device reserve battery power for more user-oriented tasks and extend the battery lifetime.

Local routing capability: Because the wireless transmission range of nodes is often limited, sources and destinations may typically not be in direct transmission range of each other. Hence, the routing protocol must

be able to discover multihop routes between sources and destinations so that communication between those nodes is possible.

Dynamic topology maintenance: Once a route is established, it is likely that some link in the route will break due to node movement. In order for a source to communicate with a destination, a viable routing path must be maintained, even while the intermediate nodes, or even the source or destination nodes, are moving. Further, because link breaks on ad hoc networks are common, link breaks must be handled quickly with a minimum of associated overhead.

Loop prevention: Routing loops occur when some node along a path selects a next hop to the destination is also a node that occurred earlier in the path. When a routing loop exists, data and control packets may traverse the path multiple times until either the path is fixed and the loop is eliminated, or until the time to live (TTL) of the packet reaches zero. Because bandwidth is scarce and packet processing and forwarding is expensive, routing loops are extremely wasteful of resources and are detrimental to the network. Even a transitory routing loop will have a negative impact on the network. Hence, loops should be avoided at all times.

To achieve availability, routing protocols[1],[2],[3] should be robust against both dynamically changing topology and malicious attacks described in .All routing protocols for ad hoc networks which mentioned in need to perform a set of basic functions in the form of route identification and route reconfiguration. Route reconfiguration functions are invoked to recover from the effects of undesirable events, such as host or link failures of various kinds, and traffic congestions appearing within a sub network. Unlike the wired networks that typically have fixed network topologies, each node in a wireless network can potentially change the network topology by adjusting its transmission range and/or selecting specific nodes to forward its messages, thus controlling its set of neighbors. The primary goal of topology control in wireless networks is to maintain network connectivity, optimize network lifetime and throughput, and make it possible to design power-efficient routing.

A route discovery protocol based on flooding can be improved. When a node S needs to find a route to node D, node S broadcasts a route request message to all its neighbors, on receiving a route request message, compares the desired destination with its own identifier. If there is a match, it means that the request is for a route to itself (i.e., node X). Otherwise, node X broadcasts the request to its neighbors – to avoid redundant transmissions of route requests, a node X only broadcasts a particular route request once (repeated reception of a route request is detected using sequence numbers).

In ad hoc networks, network topology may change instantaneously because of mobility. Figure 2.1 shows Topology change in ad hoc networks: nodes A, B, C, D, E, and F constitute an ad hoc network. The circle represents the radio range of node A. The network initially has the topology in (a). When node D moves out of the radio range of A, the network topology changes to the one in (b).

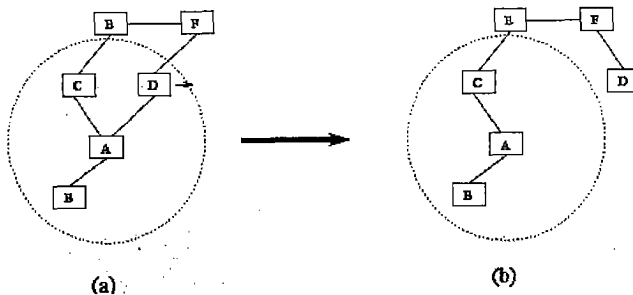


Figure 2.1

2.3 Types of Routing Protocols

Routing protocols in Ad hoc networks can be classified into three categories proactive, on-demand also called reactive, and hybrid protocols [1], [2], [3], [16].

The primary characteristic of proactive approaches is that each node in the network maintains a route to every other node in the network at all times. Route creation and maintenance is accomplished through some combination of

periodic and event-triggered routing updates. Periodic updates consist of routing information exchanges between nodes at set time intervals. Each node maintains the necessary routing information, and each node is responsible for propagating topology updates in response to instantaneous connectivity changes in the network. Examples of such protocols include those based on Destination-Sequenced Distance-Vector (DSDV) [2], [3] routing and its derivatives such as Cluster head Gateway Switch Routing (CSGR), or various link-state routing approaches, such as the Wireless Routing Protocol (WRP), STAR and Fisheye State Routing (FSR). In Reactive routing techniques, also called on-demand routing, routes are only discovered when they are actually needed. When a source node needs to send data packets to some destination, it checks its route table to determine whether it has a route. If no route exists, it performs a route discovery procedure to find a path to the destination. Hence, route discovery becomes on-demand. Examples of on-demand Protocols include Dynamic Source Routing (DSR), Temporally Ordered Routing Algorithm (TORA), and Ad hoc On-demand Distance Vector (AODV) routing [1], [2], [5] and Associativity-Based Routing (ABR). The characteristics of proactive and reactive routing protocols can be integrated in various ways to form hybrid networking protocols. Example of hybrid protocol is Zone Routing Protocol (ZRP).

2.3.1 Flooding

Flooding is the basic simplest routing method. It is best routing technique for route discovery for finding the destination and used in Dynamic source routing protocol in route discovery phase.[4],[5].

- Sender S broadcasts data packet P to all its neighbors
- Each node receiving P forwards P to its neighbors
- Sequence numbers used to avoid the possibility of forwarding the same packet more than once
- Packet P reaches destination D provided that D is reachable from sender S
- Node D does not forward the packet
- The advantages of flooding is its Simplicity

- May be more efficient than other protocols when rate of information transmission is low enough that the overhead of explicit route discovery/maintenance incurred by other protocols is relatively higher
- Many protocols perform (potentially *limited*) flooding of control packets, instead of data packets. The control packets are used to discover routes. Discovered routes are subsequently used to send data packet(s). Overhead of control packet flooding is amortized over data packets transmitted between consecutive control packet floods
- If a packet can be delivered, it will (probably multiple times).
- Since flooding naturally utilizes every path through the network, it will also use the shortest path.
- This algorithm is very simple to implement.

2.3.2 Proactive Routing protocols

The proactive routing approaches designed for ad hoc networks are derived from the traditional distance vector and link state protocols developed for use in the wire line Internet [3], [4], [5] and [13]. The primary characteristic of proactive approaches is that each node in the network maintains a route to every other node in the network at all times. Route creation and maintenance is accomplished through some combination of periodic and event-triggered routing updates. Periodic updates consist of routing information exchanges between nodes at set time intervals. The updates occur at specific intervals, regardless of the mobility and traffic characteristics of the network. Event-triggered updates, on the other hand, are transmitted whenever some event, such as a link addition or removal, occurs. The mobility rate directly impacts the frequency of event-triggered updates because link changes are more likely to occur as mobility increases.

OLSR [2], [3], [4] is a proactive routing protocol, providing the advantage of having routes immediately available in each node for all destinations in the network. The key idea of OLSR is the use of multipoint relay (MPR) nodes to flood the network in an efficient way by reducing duplicate packets in the same region. Each node i selects, independently from the other nodes, a minimal set of multipoint relay nodes, denoted as $\text{MPR}(i)$, from among its one-hop

neighbors. The nodes in MPR (i) have the following property: every node in the symmetric two hops neighbor hood of i must have a symmetric link towards MPR (i). In other words, the union of the one-hop neighbor set of MPR (i) contains the whole two-hop neighbor set. The MPR sets permit flooding to be realized efficiently: when a node i want to flood a message, it sends the message only to the nodes in MPR (i), which in turn send the message to their MPR nodes and so on. The *Multipoint Relay Selector* set (MPR selector set) of a node j is composed of the set of neighbors that have selected it as MPR. Each node periodically floods its MPR selector set, using the flooding technique and a special type of control message called a Topology Control (TC) message. Using TC messages, a node announces to the network that it has reachability to the nodes of its MPR selector set (it is its last-hop node). A TC message is stamped with a sequence number, incremented when the MPR selector set changes. The two main OLSR functionalities are neighbor discovery and topology dissemination.

The use of MPRs also minimizes flooding of control traffic. Indeed only multipoint relays forward control messages.

2.3.3 On-demand routing protocols

Reactive routing techniques, also called *on-demand* routing, take a very different approach to routing than proactive protocols. Reactive routing approaches take a departure from traditional Internet routing approaches by not continuously maintaining a route between all pairs of network nodes. Instead, routes are only discovered when they are actually needed [12], [17]. When a source node needs to send data packets to some destination, it checks its route table to determine whether it has a route. If no route exists, it performs a *route discovery* procedure to find a path to the destination. Hence, route discovery becomes on-demand. If two nodes never need to talk to each other, then they do not need to utilize their resources maintaining a path between each other. The route discovery typically consists of the network-wide flooding of a request message. On-demand routing protocols in particular, have been widely developed because they consume much less bandwidth than proactive protocols. Ad hoc On- Demand Distance Vector (AODV) [5] and Dynamic Source

Routing (DSR) are probably the two most widely studied on-demand ad hoc routing protocols [6], [11], [17].

In DSR protocol, when a node has a packet to transmit to another node, it checks its Route Cache for a source route to the destination. If there is already an available route, then the source node will just use that route immediately. If there is more than one source route, the source node will choose the route with the shortest hop-count, 'Source Route' at the S includes list of all intermediate traversing nodes in the packet, when it desires to send the packet to D in an ad hoc network. S initiates route discovery, if there are no routes in its cache. Each route request may discover multiple routes and all routes are cached at the source node. The Route Reply packet is sent back by the destination to the source by reversing the received node list accumulated in the Route Request packet. The reversed node list forms the 'Source Route' for the Route Reply packet. DSR design includes loop free discovery of routes and discovering multiple paths because paths are stored in cache, chapter3 describes more details about DSR on-demand routing protocol.

AODV attempts to improve on DSR by maintaining routing tables at the nodes, so that data packets do not have to contain routes Route Requests (RREQ) are forwarded in a manner similar to DSR When the intended destination receives a Route Request, it replies by sending a Route Reply. Route Reply travels along the reverse path set-up when Route Request is forwarded.

2.4 Multipath Routing in MANET

A critical issue in the design of ad hoc network is the development of efficient routing protocols that can provide high quality communication for each data session and especially in the presence of a large amount of traffic. Several routing protocols have already been proposed. However, most of them focus only on single path routing.

Multipath routing allows the establishment of multiple paths between a single source and single destination node [7], [8], [9], [12], [19]. It is also beneficial to avoid traffic congestion and frequent link breaks in communication because of the mobility of nodes.

3. Dynamic Source Routing protocol (DSR)

Dynamic Source Routing protocol is one of the mostly used reactive routing protocols in Mobile ad hoc networks. Our proposed routing protocol is based on DSR protocol.

3.1 DSR Protocol

The distinguishing features of the Dynamic Source Routing protocol (DSR) are [1],[2],[3],[6] [12], [20]:

- Packet forwarding via source routing
- Aggressive use of route caches that store full paths to destinations

When node S wants to send a packet to node D, but does not know a route to D, node S initiates a route discovery.

Source routing presents the following advantages:

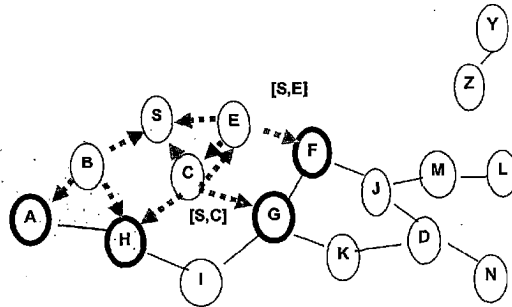
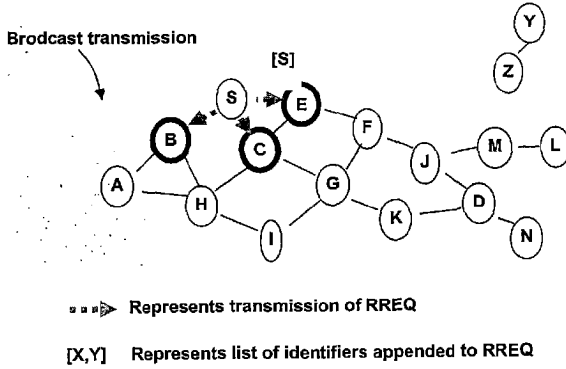
- It allows packet routing to be trivially loop-free.
- It avoids the need for up-to-date routing information in the intermediate nodes through which packets are forwarded.
- It allows nodes to cache route information by overhearing data packets.

The DSR protocol is composed of two main mechanisms, Route Discovery and Route Maintenance. Route Discovery adopts route request (RREQ)–route reply (RREP) control packets and is triggered by a node S, which attempts to send a packet to a destination node D and does not have a path into its cache. Discovery is based on flooding the network with a RREQ packet, which includes the following fields: the sender address, the target address, a unique number to identify the request, and a route record.

3.2 Route discovery

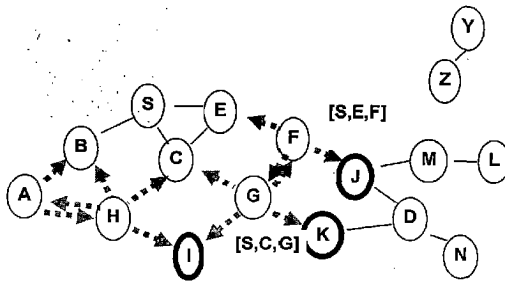
A route discovery protocol based on flooding can be improved. When a node S needs to find a route to node D, node S broadcasts a route request message to all its neighbors, on receiving a route request message, compares the desired destination with its own identifier. If there is a match, it means that the request is for a route to itself (i.e., node X). Otherwise, node X broadcasts the request to its

neighbors -- to avoid redundant transmissions of route requests, a node X only broadcasts a particular route request once (repeated reception of a route request is detected using sequence numbers).

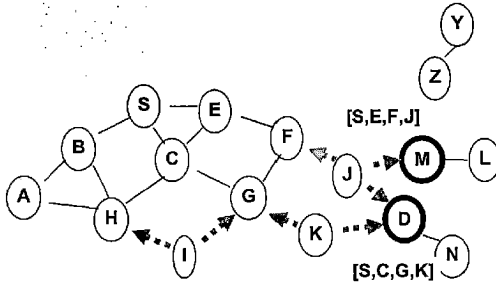


S





- Node C receives RREQ from G and H, but does not forward it again, because node C has already forwarded RREQ once



- Nodes J and K both broadcast RREQ to node D
- Since nodes J and K are hidden from each other, their transmissions may collide

Figure: 2.1 route discovery method

On receiving a RREQ control packet, an intermediate node can reply to S with a RREP if a path to the destination is stored in its cache (in this case the returned path is the concatenation of the path accumulated from S to the node and the path in the cache from the node to D), Discard the packet, if already received or it can append its own ID into the route record and broadcast the packet to its neighbors. On receiving a RREQ packet, the destination replies for request packet to S with a RREP packet. An RREP packet is routed through the path obtained by reversing the route stored in the RREQ's route record (the links are assumed bidirectional) and containing the accumulated path. Figure 2.1 shows how the route discovery method for dynamic source routing protocol in ad hoc networks.

3.3 Route maintenance

When originating or forwarding a packet using a source route, each node transmitting the packet is responsible for confirming that data can flow over the link from that node to the next hop. An acknowledgment can provide confirmation that a link is capable of carrying data, and in wireless networks, acknowledgments are often provided at no cost, as an existing standard part of the MAC protocol in use. After the acknowledgment request has been retransmitted the maximum number of times, if no acknowledgment has been received, then the sender treats the link to this next-hop destination as currently "broken".

The base mechanism for route maintenance is as follows. When an intermediate node detects that the link to its next-hop node toward the destination is broken, it removes this link from its route cache and returns a Route Error control message to S. The source S then triggers a new route discovery.

3.4 Route Caching in Dynamic Source Routing Protocol

When node S learns that a route to node D is broken, it uses another route from its local cache, if such a route to D exists in its cache. Otherwise, node S initiates route discovery by sending a route request. DSR aggressively uses route caching [6], [11]. By virtue of source routing, it is possible to cache every overheard route without causing loops. Any forwarding node caches any source

route in a packet it forwards for possible future use. Also, the destination replies to *all* requests. Thus the source learns many alternate routes to the destination that are cached. Alternate routes are useful in case the primary (shortest) route breaks. In addition, any intermediate node on a route learns routes to the source and destination as well as other intermediate nodes on that route. Thus, a large amount of routing information is gathered and cached with just a single query-reply cycle. These cached routes may be used in replying to subsequent route queries.

Using route cache has some advantages and disadvantages which includes

1. Advantages using route cache

- a. Can speed up route discovery
- b. Can reduce propagation of route requests

2. Disadvantages of route caching

- a. Stale caches can adversely affect performance
- b. With passage of time and host mobility, cached routes may become invalid
- c. A sender host may try several stale routes (obtained from local cache, or replied from cache by other nodes), before finding a good route

3.5 Dynamic Source Routing Protocol: Advantages and Disadvantages

Advantages:

1. Routes maintained only between nodes that need to communicate
 - a. reduces overhead of route maintenance
2. Route caching can further reduce route discovery overhead
3. A single route discovery may yield many routes to the destination, due to intermediate nodes replying from local caches

Disadvantages:

1. Packet header size grows with route length due to source routing
2. Flood of route requests may potentially reach all nodes in the network
3. Care must be taken to avoid collisions between route requests propagated by neighboring nodes insertion of random delays before forwarding RREQ
4. Increased contention if too many route replies come back due to nodes replying using their local cache

- a. Route Reply *Storm* problem
 - b. Reply storm may be eased by preventing a node from sending RREP if it hears another RREP with a shorter route
5. An intermediate node may send Route Reply using a stale cached route, thus polluting other caches

3.6 Additional Route Discovery Features

- **Caching Overheard Routing Information**

A node forwarding or otherwise overhearing any packet should add usable routing information from that packet to its own Route Cache.

- **Replying to Route Requests using Cached Routes**

A node receiving a Route Request, for which it is not the target, searches its own Route Cache for a route to the target of the Request. If found, the node generally returns a Route Reply to the initiator itself rather than forwarding the Route Request.

- **Preventing Route Reply storms**

In order to reduce multiple route replies to be simultaneously transmitted back to the source, if a node can put its network interface into promiscuous receive mode, it MAY delay sending its own Route Reply for a short period, while listening to see if the initiating node begins using a shorter route first. Specifically, this node MAY delay sending its own Route Reply for a random period.

3.7 Additional Route Maintenance features

- **Packet salvaging**

When an intermediate node finds the route to the destination to be broken, instead of discarding the packet, it “salvages” it by sending it in an alternate route that it retrieves from its route cache (if present).

4. Load balancing in ad hoc networks

Load balancing has been one of the major research interests in both wired and wireless networks[7],[8],[9],[10]. Mobile Ad hoc networks with nodes having different processing powers and thus can perform extensive computations apart from forwarding packets for other nodes. These nodes will also have various degrees of battery powers as well. Due to the heterogeneity of the systems in terms of processing and battery powers, naturally, there will be load imbalance. If the workload is distributed among the nodes in the system based on the resources of individual nodes, the average execution time can be minimized and the lifetime of the nodes can be maximized.

Congestion occurs when the amount of data sent to the network exceeds the available capacity. Such situation leads to increased buffer space usage in intermediate nodes over the data path, leading to data losses in case of shortage of resources.

Mobility, channel error, and congestion are the main causes for packet loss in mobile ad hoc networks [8], [9], and [10]. Reducing packet loss typically involves congestion control operating on top of a mobility and failure adaptive routing protocol at the network layer.

Routing protocols can be categorized into two types, Congestion-aware routing and congestion adaptive routing protocols [9]. In congestion-aware routing techniques, congestion is taken into consideration only when establishing a new route which remains the same until mobility or failure results in disconnection. In congestion-adaptive routing, the route is adaptively changeable based on the congestion status of the network.

Congestion non-adaptive in routing in MANETs may lead to the following problems:

Long delay: It takes time for a congestion to be detected by the congestion control mechanism. In severe congestion situations, it may be better to use a

new route. The problem with an on-demand routing protocol is the delay it takes to search for the new route.

High overhead: In case a new route is needed, it takes processing and communication effort to discover it. If multipath routing is used, though an alternate route is readily found, it takes effort to maintain multiple paths.

Many packet losses: Many packets may have already been lost by the time congestion is detected. A typical congestion control solution will try to reduce the traffic load, either by decreasing the sending rate at the sender or dropping packets at the intermediate nodes or doing both. The consequence is a high packet loss rate or a small throughput at the receiver.

4.1 Single-path versus Multi-path Routing

Multi-path routing balances the load significantly better than single-path routing [15], [17], [18]. Instead of using a single path; multiple paths to route messages between any source-destination pair of nodes balances the load more evenly throughout the network.

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

4.2 Multipath Source Routing in Wireless Ad Hoc Networks

A multipath routing protocol MSR, which is based on DSR [18], also uses source routing. Multipath routing can increase application performance by giving applications the freedom to use multiple paths within the same path service. On the other hand, maintaining alternative paths requires more routing

table space and computation overload. However, some DSR's characteristics can suppress these disadvantages. First, Source Routing is so flexible that messages can be forwarded on arbitrary paths, which makes it very easy to dispatch messages to multiple paths without demanding path calculation in the intermediate hops. Second, the on-demand nature of DSR reduces the routing storage greatly. There are three elements necessary to make multipath network viable: (i) appropriate paths calculated between nodes, (ii) efficient packet forwarding on calculated paths, and (iii) effective end-host usage of multiple paths. The three issues in MSR are addressed as follows

4.2.1 Loop-free problem

As the route is part of the packet itself, routing loops, either short- or long-lived, cannot be formed as they can be immediately detected and eliminated.

4.2.2 Packet forwarding and load balancing

Since MSR uses source routing [18], packet forwarding in the intermediate nodes does nothing but forwarding the packet as the route in its header indicated, adding no further processing complexity than that in DSR [2],[3],[18]. All the work for path calculation is done in the source hosts. Then, for MSR, there are some works of load balancing to do in the source nodes. In our experiment, a special table containing multiple path information to the specific destination is built, as illustrated as follows.

```
struct mul-dest
{
int index ;
ID Dest;
float Delay;
float Weight;
...};
```

Dest is the destination of a route. *Index* is the current index of the route in DSRs route cache that has a destination to *Dest*. *Delay* is the current estimate of the round-trip time. *Weight* is a per-destination based load distribution weight between all the routes

that have the same destination. *Weight* is in terms of the number of packets to be sent consecutively on d_i , the same route every time. We choose the weight W_i^j (i is the index of the route to j) according to a heuristic equation given below,

$$W_i^j = \text{Min}_j \left(\left[\frac{d_{\max}^j}{d_i^j} \right], U \right) \times R$$

Where d_{\max}^j is the maximum delay of all the routes to the same destination, d_i^j is the delay of route with index i , and R is a factor to control the switching frequency between routes. U is a bound value to insure that W_i^j should not be too large. The larger the R 's value, the less frequently the switching happens and the less processing overload of searching and positioning an entry in the multistage table. When choosing R , the IFQ's buffer's size should also be taken into considerations. In our experiment (IFQ size is 50), R is set to 1. To aid the load balancing, a probing mechanism is employed. An RTT measurement tool for DSR and MSR in simulation, **SRping** is developed to get the RTT between two arbitrary nodes. When distributing the load, the weighted-round-robin scheduling strategy is used. Probing is also an enhancement to the DSR Route maintenance mechanism. Normally, in **DSR**, a link breakage can be notified only when a Route Error message is returned. However, in wireless mobile environment, it has a nontrivial chance that the Route Error message can not reach the original sender successfully. Although, "as a last resort, a bit in the packet header could be included to allow a host **transmitting** a packet to request an explicit acknowledgement from the next-hop receiver", probing one path constantly only to test its validity is not cost effective.

4.3 Proposed Adaptive Multipath Source Routing Protocol

Congestion at the links and in the routers is the main cause of large delays in the Internet; the same is true in ad hoc networks where bandwidths are always very limited. Routing protocols used in conventional wired networks (e.g., Bellman-Ford and link state) [2],[3] are not well suited for the mobile environment due to the considerable overhead produced by periodic route update messages, and to their slow convergence to topological changes. In addition, all the Internet routing protocols in use today rely on single-path routing algorithms, which not

only under-utilize resources, but also cannot cope with congestion and link breakage. This can be attributed to the fact that all traffic for a destination is required to be routed through a single successor. So when a link becomes congested or broken, its entire carried traffic has to be rerouted; this becomes more time consuming in mobile networks. If link costs are made functions of congestion or delays, routing table entries can become unstable in single-path routing protocols.

Multipath routing can overcome the above problem. In addition, it can provide load balancing and route failure protection by distributing traffic among a set of diverse paths. These benefits make multipath routing a good candidate for bandwidth limited and mobile ad hoc networks. However, maintaining alternative paths requires much more routing table space and computation, especially for table-driven routing algorithms in large networks. Many ad hoc routing protocols have been proposed recently, such as AODV, DSDV, DSR, and TORA. However, they are all single-path based. DSR is capable of reducing communication and computation overhead. Our proposed protocol is based on the on-demand routing protocol called Dynamic Source Routing protocol.

Our motivation is that congestion is a dominant cause for packet loss in MANETs. Unlike well-established networks such as the Internet, in a dynamic network like a MANET, it is expensive, in terms of time and overhead, to recover from congestion. Our proposed CCSR protocol tries to prevent congestion from occurring in the first place. CCSR uses congestion status of the whole path (Congestion Status of the all nodes participated in route path) and source node maintains the table called Congestion Status table (Cst) contains the congestion status of the every path from source node to destination node.

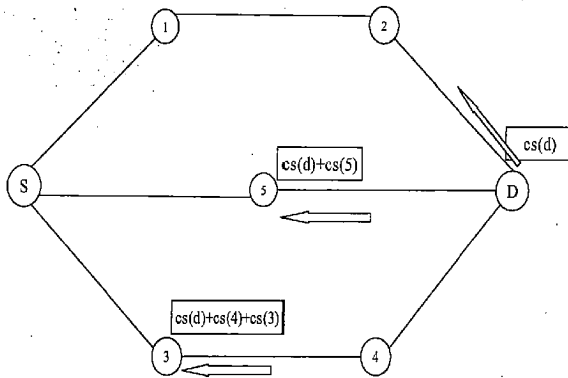


Fig 1.

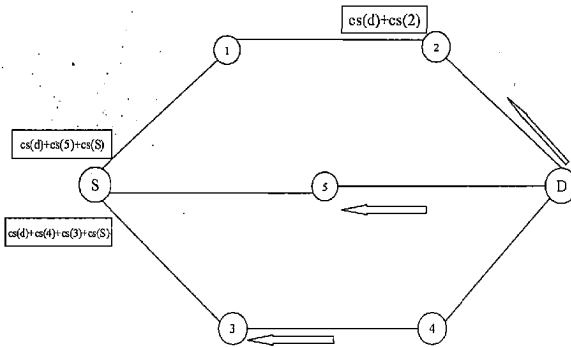


Fig 2

A simplified example is illustrated in Fig. 1. Three possible routes $S \rightarrow 1 \rightarrow 2 \rightarrow D$, $S \rightarrow 5 \rightarrow D$ and $S \rightarrow 3 \rightarrow 4 \rightarrow D$ are multipath routes between source node S to the destination node D. Source node S maintains a special table called Congestion Status Table, which stores the congestion status of the every path, remember that here we are calculating the congestion status not for the single node rather all nodes of the path (cumulative congestion status).

In CCSR Destination will send Cumulative congestion status packets (Cosp) packets periodically towards the source node. Source node after receiving the Cosp packets it will update the Cst Table. According to the Cst table Source

node will distribute the packets such that more packet towards the path with less congestion status and sends less packets to the path with more congestion status in the Cst table. The Congestion Status of the path will be calculated as,

Cs (A): indicates Congestion status of the node A.

Ccs(B): indicates Cumulative Congestion Status of the node B. calculated using the congestion status of the node B plus congestion status of its previous nodes.

Here Congestion Status of the particular node will be calculated using available buffer size or queue length and number of packets, the ratio of data to the available queue length will give the congestion status of the particular node.

When the number of packets coming to a node exceeds its carrying capacity, the node becomes congested and starts losing packets. We can use a variety of metrics at a node to monitor congestion status. For instance, we can be based on the percentage of all packets discarded for lack of buffer space, the average queue length; the number of packets timed out and retransmitted the average packet delay, and the standard deviation of packet delay. In all cases, rising numbers indicate growing congestion. The design of CCSR allows it to work on top of any of these methods. For ease of presentation and as a proof of concept, we adopt the following simple method as an example in the paper. A node periodically checks the occupancy of its link-layer buffer. The congestion status is determined based on the ratio r between the number of packets currently buffered to the buffer size.

Primary Route Discovery: The sender discovers the route to the receiver in a simple way. It broadcasts an REQ packet toward the receiver. The receiver responds to the copy of REQ by sending back an REP packet. The REP will traverse back the path that the REQ previously followed. The over head of the replay packets can be reduced for reducing the over head of the protocol. Rather than sending the replay packets to every REQ packets destination can use the hop count and will send the replay packet only if the hop count is less than previous one. This path becomes the primary route between the sender and the receiver.

The cumulative congestion status of the path will be calculated as,

Ccs (D): Cumulative Congestion Status of the node D of the path $\{S, 1, 2, D\}$

=Congestion Status of the node D

Ccs (1): Cumulative Congestion Status of the node 1 of the path {S, 1, 2, D}.

=Congestion Status of the node D + Congestion Status of the node 1

= C_s (D) + C_s (1).

Ccs (3): Cumulative Congestion Status of the node 3 of the path {S, 3, 4, D}.

=C_s (D) + C_s (4) + C_s (3).

The typical Cst table of the source node S is shown in the table 1, where pathID indicates the nodes involved in routing and Congestion Status indicates the cumulative congestion status of all nodes involved in the route path. After updating the latest congestion status source node will choose the path and distribute the packets. Load distribution procedure is shown below .

/*

a, b and c indicates the number packets available at corresponding paths A, B and C and x, y and z indicates queue length of the paths then,

a/x, b/y and c/z indicates the congestion status of the paths

NOPACK is data available at source node

*/

Procedure LoadDIST (NOPACK, A, B, C)

Repeat until NOPACK = 0

If C_s (A) or C_s (B) or C_s(C) = max {x or y or z}

Stop sending packets toward the path

Else if C_s (C) > C_s (A) and C_s(C) >C_s (B)

Send (C_s(C)) / (C_s (A)) packets to the path A,

(C_s(C)) / (C_s (B)) packets to the path B,

(C_s(C))/(C_s(C)), {1} packet to the path C

End

End

Path ID	Congestion Status
{S,1,2,D}	Cs(S+1+2+D)
{S,5,D}	Cs(S+5+D)
{S,3,4,D}	Cs(S+3+4+D)
.....
.....

Table 1

4.4 Adaptive multipath source routing protocol: advantages and disadvantages

End-to-end delay: Consistently in simulation runs, proposed protocol provided an average delay shorter than did AODV and DSR [5], [6].

Data packet delivery ratio: Both CCSR and AODV successfully delivered more data packets than DSR. However, when the network was heavily loaded, whether the network was steady or highly mobile, CCSR performed better than AODV.

Energy efficiency: CCSR and AODV were consistently better than DSR. CRP was more efficient than AODV, especially when the network traffic was heavier.

Because CCSR protocol is based on multipath routing protocol, maintaining multiple paths rather than single path is difficult task. Overhead of these paths maintaining and controlling is one of the disadvantages of this multipath routing protocol. Searching and sorting methods are introduced and the source node is the responsible for this task, time consuming for searching in the Congestion status table at source node is another disadvantage for this protocol.

5. The Network Simulator

5.1 GLOMOSIM Overview

Global Mobile Information System Simulator (GloMoSim) is a scalable simulation environment for large wireless communication networks [16].

GloMoSim simulates networks with up to thousand nodes linked by a heterogeneous communications capability that includes multicast, asymmetric communications using direct satellite broadcasts, multi-hop wireless communications using ad-hoc networking, and traditional Internet protocols

5.2 Use of GloMoSim Simulator

After installing GloMoSim, a simulation can be started by executing the following command in the *BIN* subdirectory.

```
./glomosim < input file >
```

The <*input file*> contains the configuration parameters for the simulation (an example of such file is CONFIG.IN). A file called GLOMO.STAT is produced at the end of the simulation and contains all the statistics generated.

5.3 Basic Structure of the source directory

GloMoSim contains following directories [16]

/doc – contains the documentation

/scenarios- contains directories of various sample configuration topologies

/main - contains the basic framework design

/bin- for executable and input/output files

/include- contains common include files

/application- contains code for the application layer

/transport -contains the code for the transport layer

/network -contains the code for the network layer

/Mac -contains the code for the Mac layer

/radio- contains the code for the physical layer

5.4 The Application Configuration File

Applications such as FTP and Telnet are configured in this file. The traffic generators currently available are FTP, FTP/GENERIC, TELNET, CBR, and HTTP. FTP uses tcplib to simulate the file transfer protocol.

In order to use FTP, the following format is needed:

FTP <src> <dest> <items to send> <start time>

Where <src> is the client node, <dest> is the server node, <items to send> is how many application layer items to send, and <start time> is when to start FTP during the simulation. If <items to send> is set to 0, FTP will use tcplib to randomly determine the amount of application layer items to send. The size of each item will always be randomly determined by tcplib.

Examples:

- FTP 0 1 10 0S. Node 0 sends node 1 ten items at the start of the simulation, with the size of each item randomly determined by tcplib.
- FTP 0 1 0 100S. Node 0 sends node 1 the number of items randomly picked by tcplib after 100 seconds into the simulation. The size of each item is also randomly determined by tcplib.

FTP/GENERIC does not use tcplib to simulate file transfer. Instead, the client simply sends the data items to the server without the server sending any control information back to the client.

FTP/GENERIC, the following format is needed:

**FTP/GENERIC <src> <dest> <items to send> <item size> <start time>
<end time>**

where <src> is the client node, <dest> is the server node, <items to send> is how many application layer items to send, <item size> is size of each application layer item, <start time> is when to start FTP/GENERIC during the simulation, and <end time> is when to terminate FTP/GENERIC. If <items to send> is set to 0, FTP/GENERIC will run until the specified <end time> or until the end of the simulation, whichever comes first. If <end time> is set to 0, FTP/GENERIC will run until all <items to send> is transmitted or until the end

of simulation, whichever comes first. If <items to send> and <end time> are both greater than 0, FTP/GENERIC will run until either <items to send> is done, <end time> is reached, or the simulation ends, whichever comes first.

Examples:

- FTP/GENERIC 0 1 10 1460 0S 600S. Node 0 sends node 1 ten items of 1460B each at the start of the simulation up to 600 seconds into the simulation. If the ten items are sent before 600 seconds elapsed, no other items are sent.
- FTP/GENERIC 0 1 10 1460 0S 0S. Node 0 sends node 1 ten items of 1460B each at the start of the simulation until the end of the simulation. If the ten items are sent the simulation ends, no other items are sent.
- FTP/GENERIC 0 1 0 1460 0S 0S. Node 0 continuously sends node 1 items of 1460B each at the start of the simulation until the end of the simulation

TELNET uses `tcplib` to simulate the telnet protocol. In order to use TELNET, the following format is needed:

TELNET <src> <dest> <session duration> <start time>

Where <src> is the client node, <dest> is the server node, <session duration> is how long the telnet session will last, <start time> is when to start TELNET during the simulation. If <session duration> is set to 0, TELNET will use `tcplib` to randomly determine how long the telnet session will last. The interval between telnet items are determined by `tcplib`.

Examples:

- TELNET 0 1 100S 0S. Node 0 sends node 1 telnet traffic for a duration of 100 seconds at the start of the simulation.
- TELNET 0 1 0S 0S. Node 0 sends node 1 telnet traffic for a duration randomly determined by `tcplib` at the start of the simulation.

CBR simulates a constant bit rate generator. In order to use CBR, the following format is needed:

**CBR <src> <dest> <items to send> <item size> <interval> <start time>
<end time>**

Where <src> is the client node, <dest> is the server node, <items to send> is how many application layer items to send, <item size> is the size of each application layer item, <interval> is the inter-departure time between the application layer items, <start time> is when to start CBR during the simulation, <end time> is when to terminate CBR during the simulation. If <items to send> is set to 0, CBR will run until the specified <end time> or until the end of the simulation, which ever comes first. If <end time> is set to 0, CBR will run until all <items to send> is transmitted or until the end of simulation, which ever comes first. If <items to send> and <end time> are both greater than 0, CBR will run until either <items to send> is done, <end time> is reached, or the simulation ends, which ever comes first.

Examples:

- CBR 0 1 10 1460 1S 0S 600S. Node 0 sends node 1 ten items of 1460B each at the start of the simulation up to 600 seconds into the simulation. The inter-departure time for each item is 1 second. If the ten items are sent before 600 seconds elapsed, no other items are sent.
- CBR 0 1 0 1460 1S 0S 600S. Node 0 continuously sends node 1 items of 1460B each at the start of the simulation up to 600 seconds into the simulation. The inter-departure time for each item is 1 second.
- CBR 0 1 0 1460 1S 0S 0S. Node 0 continuously sends node 1 items of 1460B each at the start of the simulation up to the end of the simulation. The inter-departure time for each item is 1 second.

HTTP simulates single-TCP connection web servers and clients. The following format describes its use for servers:

HTTPD <address>

Where <address> is the node address of a node which will be serving Web pages. For HTTP clients, the following format is used:

**HTTP <address> <num_of_server> <server_1> ... <server_n> <start>
<thresh>**

Where <address> is the node address of the node on which this client resides, <num of server> is the number of server addresses which will follow <server 1>, <server n> are the node addresses of the servers which this client will choose between when requesting pages. There must be HTTPD <address> lines existing separately for each of these addresses; <start> is the start time for when the client will begin requesting pages <thresh> is a ceiling (specified in units of time) on the amount of *think time* that will be allowed for a client. The network-trace based amount of time modulo this threshold is used to determine think time.

Example:

HTTPD 2

HTTPD 5

HTTPD 8

HTTPD 11

HTTP 1 3 2 5 11 10S 120S

There are HTTP servers on nodes 2, 5, 8, and 11. There is an HTTP client on node 1. This client chooses between servers only when requesting web pages. It begins browsing after 10S of simulation time have passed, and will *think* (remain idle) for at most 2 minutes of simulation time, at a time.

5.5 CONFIG.IN FILE

During the simulation takes the input from the file called config.in, which specifies various parameters to be used.

1. Simulation time: Maximum simulation time. The number portion can be followed by optional letters to modify the simulation time. For example, 100NS (100 nano-seconds), 100MS (100 milli-seconds), 100S or 100 (100 seconds), 100M (100 minutes), 100H (100 hours) and 100D (100 days)

2. A random number seed used to initialize part of the seed of various randomly generated numbers in the simulation. This can be used to vary the seed of the simulation to see the consistency of the results of the simulation.
3. Parameters stand for the physical terrain in which the nodes are being simulated. For example, the following represents an area of size 100meters by 100 meters. All range parameters are in terms of meters.
4. The number of nodes being simulated.
5. Parameter representing the node placement strategy.

RANDOM: Nodes are placed randomly within the physical terrain.

UNIFORM: Based on the number of nodes in the simulation, the physical terrain is divided into a number of cells. Within each cell, a node is placed randomly.

FILE: Position of nodes is read from NODE-PLACEMENT-FILE. On each line of the file, the x and y position of a single node is separated by a space.

6. Parameters for mobility.

If MOBILITY is set to NO, than there is no movement of nodes in the model. RANDOM-DRUNKEN model, if a node is currently at position (x, y), it can possibly move to (x-1, y),(x+1, y), (x, y-1), and (x, y+1); as long as the new position is within the physical terrain.

RANDOM WAYPOINT, a node randomly selects a destination from the physical terrain. It moves in the direction of the destination in a speed uniformly chosen between MOBILITY-WP-MIN-SPEED and MOBILITY-WP-MAX-SPEED (meter/sec). After it reaches its destination, the node stays there for MOBILITY-WP-PAUSE time period.

The MOBILITY-INTERVAL is used in some models that a node updates its position every MOBILITY-INTERVAL time period. The MOBILITY-D-UPDATE is used that a node updates its position based on the distance (in meters).

7. Propagation-limit:

Signals with powers below PROPAGATION-LIMIT (in dB) are not delivered. This value must be smaller than RADIO-RX-SENSITIVITY + RADIO-

ANTENNA-GAIN of any node in the model. Otherwise, simulation results may be incorrect. Lower value should make the simulation more precise, but it also makes the execution time longer.

8. Propagation-pathloss: pathloss model

FREE-SPACE: Friss free space model.

(Path loss exponent, sigma) = (2.0, 0.0)

TWO-RAY: It uses free space path loss (2.0, 0.0) for near sight and plane earth path loss (4.0, 0.0) for far sight.

9. NOISE-FIGURE: noise figure

10. TEMPERATURE of the environment

11. RADIO-TYPE:

It represents the radio model to transmit and receive packets. It may be one of the following

RADIO-ACCNOISE: standard radio model

RADIO-NONNOISE: abstract radio model

12. RADIO-FREQUENCY: frequency (in hertz)

13. RADIO-BANDWIDTH: bandwidth (in bits per second)

14. RADIO-RX-TYPE: packet reception model

15. RADIO-TX-POWER: radio transmission power (in dBm)

16. RADIO-ANTENNA-GAIN: antenna gain (in dB)

17. RADIO-RX-SENSITIVITY: sensitivity of the radio (in dBm)

18. RADIO-RX-THRESHOLD: Minimum power for received packet (in dBm)

19. Protocol to be used in MAC layer

20. Parameter to enable (Or) disable the PROMISCOUS mode

21. Protocol to be used in NETWORK layer

22. ROUTING-PROTOCOL to be used

23. Input file to setup applications such as FTP and Telnet.

6. Results and Discussion

This CCSR protocol was simulated in GloMoSim Network Simulator. Number of nodes present in the network was varied from 20 to 60. Nodes moved in an area of (1000x300) m² in accordance with random waypoint mobility model, with a velocity of 20 m/s and a pause time of 0 second. Simulation time was set as 700 seconds.

6.1 Performance Metrics

We considered the following important metrics for the evaluation:

1. Data throughput
2. End-to-end delay
3. Energy conception

Data Throughput (kilobits per second –Kbps) - describes the average number of bits received successfully at the destination per unit time (second). This metric was chosen to measure the resulting network capacity in the experiments.

End-to-end delay (seconds) – This is an average of the sum of delays (including latency), at each destination node during the route discovery from the source to destination.

We have simulated the proposed protocol in GloMoSim for different environments, every time during the simulation this protocol gives the better performance in terms of delay and packet delivery ratio of the network than the protocols Dynamic source routing protocol. We have simulated our protocol for 20,30,40,50 and 60 nodes. Figure 3 and 4 shows the simulation results of end-to-end delay and packet deliver ratio comparisons of DSR and CCSR. The performance of proposed routing protocol is improved in terms of delay and also number of packets received at destination node is increased up to 25 percentages. Using CCSR, energy also saved significantly because this protocol is based on load balancing technique. Battery power of a node is depend on processing of the number packets per unit time. The processing of the packets per unit time will reduced in CCSR protocol. Because our proposed protocol uses load balancing and congestion at the nodes of the network is prevented and

controlled. The mobility speed of the network is 10 m/s and simulation time is 600 sec and used random way point as network model.

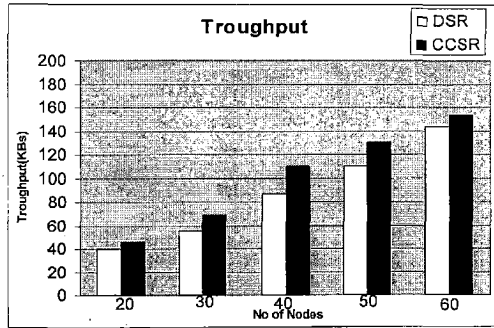


Fig 3

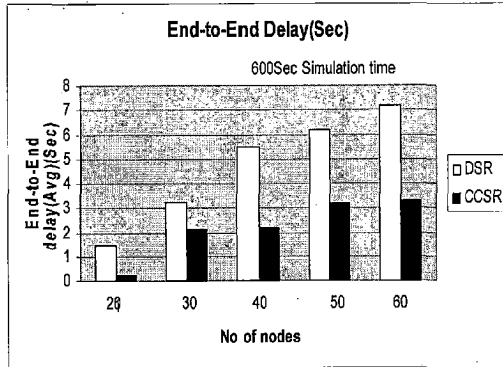


Fig 4

Figure 5 and 6 shows the simulation results of proposed routing protocol CCSR and AODV proactive routing protocol. Our CCSR protocol gives the better performance not only the DSR protocol but also AODV. We have simulated for different simulation times from 500sec to 800 sec .during the all simulation environment situations our proposed protocol produces better results in terms of packet delivery ratio and end-to end delay. Because DSR and AODV routing

protocols are suffering the problem of load unbalance and intermediate nodes are congested as a result of that delay is increasing and decreasing the packet delivery ratio. Our proposed routing protocol balances the load significantly and increases the packet delivery ratio (throughput). The mobility speed of the network is 20 m/s and simulation time is 600 sec and used random way point as network model.

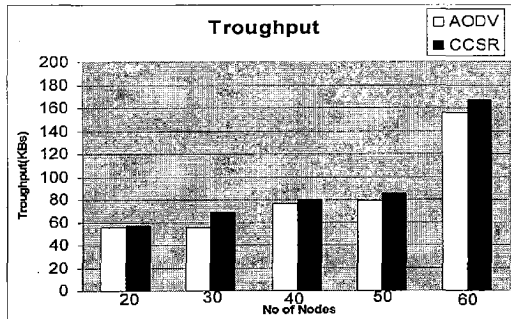


Fig. 5

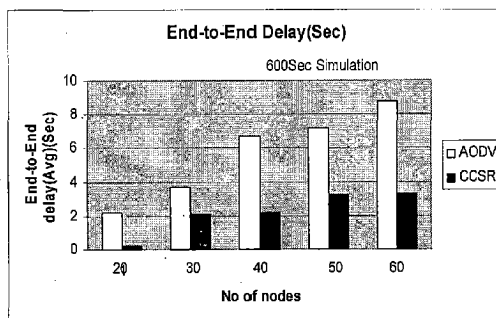


Fig. 6

7. Conclusion and future work

In this thesis, we proposed a new routing protocol model, an adaptive multipath source routing protocol for ad hoc networks. Our proposed protocol is based on congestion status and route discovery, route maintenance techniques are based on Dynamic Source routing protocol. We simulated it on different conditions (various Simulation times, number of nodes, velocity of the nodes) and determined its performance. Our proposed protocol gives better performance than Dynamic source routing protocol and Ad hoc on-demand Distance Vector Routing protocol in terms of packet delivery ratio and delay. In CCSR, load will be distributed according to the cumulative congestion status of the path. Source will distribute packets such that more packets towards the path with minimum congestion status value and will send the less packets to the path with maximum congestion value.

In future, this work can be extended in following ways:

- **Multipath Minimization:** If the path is congested frequently or the path is highly congested we can eliminate this path for decreasing the overhead of the protocol.
- **Delay as metric:** We can extend the work as, source can send the packets to the path with minimum delay and this delay can be calculated using the time stamp value. Distribute the packets according to these time stamp values. Load can be distributed according to the minimum time stamp value.
- We can extend the work as, finding the timestamp values of data packets and distribute the packets according to the time stamp values. Time stamp value indicates the delay of the packets. Maintain tables called time stamp table which contain the timestamp values (delays of the packets) of the packets. The disadvantage may be maintained a table at source node is difficult task.

References

- [1] E.Royer and C.K. Toh, "A Review of Current Routing Protocols for Ad Hoc MobileWireless Networks", IEEE Personal Communications Magazine, vol. 6, Apr. 1999.
- [2] Stefano Basagni, Marconti, Silvia Giordano and Ivan Stojmenovic, "Mobile Ad hoc Networking", IEEE Press, First Edition, 2004.
- [3] Mohammad Iliyas, "The Handbook of Ad Hoc Wireless Networks", CRC Press, First Edition, 2003.
- [4] T.Clausen and P.Jaquet, "The optimized link state routing for ad hoc network: protocol specification", RFC 3626, 2004.
- [5] C. Perkins, "Ad Hoc on Demand Distance Vector (AODV) Routing", Internet-Draft, Nov. 1997, URL://www.ietf.org/internetdrafts/draft-ietf-manet-aodv-00.txt.
- [6] David B.Johnson, David A.Maltz and Yih Chun Hu, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)", Internet Draft, Apr. 2003, URL://www.ietf.org/internetdrafts/draft-ietf-manet-dsr-09.txt.
- [7] Yashar Ganjali and Abtin Keshavarzian, "Load Balancing in Ad Hoc Networks: Single-path routing vs. Multi-path Routing", IEEE INFOCOM 2004, Twenty-third Annual Joint Conference of the IEEE Computer Communications Society, vol. 2, Mar 2004, pp. 1120-1125.
- [8] Salma Ktari, Houda Labiod and Mounir Frikha, "Load Balanced Multipath Routing in Mobile Ad hoc Networks", 10th IEEE International Conference on Communication Systems 2006 - ICCS2006, Singapore, Oct. 2006, pp.1-5.

and Xuming Fang, "Routing with Congestion Control and Load Balancing in Wireless Mesh Networks", 6th International Conference on Communications – ITS2006, pp. 719-724.

ra, R.B. Patel and V.K.Bhat, "Routing with Load Balancing in Network: A Mobile Agent Approach", 6th IEEE/ACIS 11th International Conference on Computer and Information Science (ICIS 2006), pp. 480-486.

Marina and Samir R. Das, "Performance of Route Caching in Dynamic Source Routing", International Conference on Computing Systems, Apr. 2001, pp. 425 – 432.

and S. R. Das, "On-demand Multipath routing for mobile ad hoc networks", Proc. IEEE ICCCN, Oct. 1999, pp. 64 – 70.

J.K. Toh, and M. Gerla, "Performance Evaluation of Table-Driven On-Demand Ad Hoc Routing Protocols," Proc. IEEE Symp. Indoor and Mobile Radio Comm., Sept. 1999, pp. 297-301.

Lee, Nguyen Thi Thanh Tuand Jung-Seok Heo, "Load Balancing and Route Discovery Method Based on AODV", IEEE The first International Forum on Strategic Technology, 18-20 Oct. 2006, pp. 374-378.

Li, Houde Labiod and Mounir Frikha, "Load Balanced Multipath Routing in Mobile Ad hoc Networks", 10th IEEE International Conference on Communication Systems - ICCS2006, Singapore, Oct. 2006, pp. 1-5.

Li, INRS – Universite de Quebec, A Comprehensive Tutorial, Sep. 4, 2003.

Li, R. Castaneda, and S. Das, "Performance of multipath routing and protocols in ad hoc networks," ACM/Kluwer Mobile

orks and Applications (MONET) Journal, Aug. 2001, vol. 6, pp. 339-

Yang, Lianfang Zhang, Yantai Shu and Miao Dong, "Multipath
e Routing in Wireless Ad Hoc Networks", International Conference
IEEE Computer Communications Society, Nov. 2000, pp. 479-483.

arlman, Z. Haas, P. Scholander, and S. Tabrizi, "On the impact of
ate path routing for load balancing in mobile ad hoc networks,"
International Symposium on Mobile Ad Hoc Networking and
uting (MobiHoc 2000), Aug. 2000.

am and S. Perreau, "Multi-path routing protocol with load balancing
r in mobile ad hoc network," IFIP Int'l Conference on Mobile and
ess Communications Networks (MWCN 2002), Sep. 2002.

arasi.M and Palanivelu.T.G, "A Strategy to Reduce Control Packet
of MANETs with Bidirectional Links using DSR", IEEE 9th
ational Conference on Information Technology (ICIT'06), Dec.2006,
' - 88.

INDEX

SRCE CODE LISTING

```
lude <stdlib.h>
lude <stdio.h>
lude <string.h>
lude <assert.h>
lude <math.h>s
```

rocessing procedure when Route Request is received

```
.RoutingDsrHandleRequest(GlomoNode *node, Message *msg, int ttl)

static Scount = 0;
DSR_PacketType Spkctype;
NODE_ADDR Ssrcaddr;
NODE_ADDR Stargetaddr;
int Shopcount;
NODE_ADDR Spath[DSR_MAX_SR_LEN];
int Sseqnumber;
int Scount;
GlomoNetworkIp* ipLayer = (GlomoNetworkIp *) node->networkData.networkVar;
GlomoRoutingDsr* dsr = (GlomoRoutingDsr *) ipLayer->routingProtocol;
SR_RouteRequest *rreq = (DSR_RouteRequest *)GLOMO_MsgReturnPacket(msg);
HeaderType *ipHdr = (IpHeaderType *)GLOMO_MsgReturnPacket(msg);
node->Scount=1;
if (rreq->targetAddr == node->nodeAddr) //check is it dest node?

RoutingDsrInitiateRREP(node, msg);
```



```
//node->Scount=(node->Scount)+1;
```

```
Not a destination; if the request is not seen before */  
se if (!RoutingDsrLookupRequestSeen(rreq->srcAddr,  
    rreq->seqNumber,  
    &dsr->requestSeenTable))
```

```
/* Insert request info into request seen table */  
RoutingDsrInsertRequestSeen(node,  
    rreq->srcAddr,  
    rreq->seqNumber,  
    &dsr->requestSeenTable);
```

```
/* Check if its address is in the path of the packet */  
if (!RoutingDsrCheckRequestPath(node,  
    rreq->path,  
    rreq->hopCount - 1))
```

```
{  
    /* If it has a route to destination, send a Route Reply */  
    if (RoutingDsrCheckRouteExist(rreq->targetAddr,  
        &dsr->routeCacheTable))  
    {  
        RoutingDsrInitiateRREPbyLN(node, msg);  
    } /* if check route exist */
```

```
/* Does not have any route in cache; Relay the packet if ttl > 0 */  
else if (ttl > 0 && rreq->hopCount < DSR_MAX_SR_LEN)  
{  
    RoutingDsrRelayRREQ(node, msg, ttl);  
} /* else if ttl > 0 */
```

```

else
{
    GLOMO_MsgFree(node, msg);
}
} /* if check request path */
else
{
    GLOMO_MsgFree(node, msg);
}
* else if lookup request seen */

```

```

;e

```

```

GLOMO_MsgFree(node, msg);

```

```

Handle Request */

```

stination of the route sends Route Reply in reaction to Route Request

```

RoutingDsrInitiateRREP(GlomoNode *node, Message *msg)

```

```

lomoNetworkIp* ipLayer = (GlomoNetworkIp *)node->networkData.networkVar;
lomoRoutingDsr* dsr = (GlomoRoutingDsr *)ipLayer->routingProtocol;
lessage *newMsg;
SR_RouteRequest *reqPkt;          // create request packet parameter pointer
SR_RouteReply *repPkt;           // create reply packet parameter pointer
lar *pktPtr;
it pktSize = sizeof(DSR_RouteReply); // packet size of reply packet
it i;
locktype delay;

```

```

reqPkt = (DSR_RouteRequest *) GLOMO_MsgReturnPacket(msg);

newMsg = GLOMO_MsgAlloc(node, GLOMO_MAC_LAYER, 0,
    j_MAC_FromNetwork); //node:node which is allocating message
    //layerType:Layer type to be set for this message
    //protocol:Protocol to be set for this message
    //eventType:event type to be set for this message
GLOMO_MsgPacketAlloc(node, newMsg, pktSize);
node:node which is allocating message
msg:message for which data has to be allocated
payloadSize: size of the payload to be allocated

dstPtr = (char *) GLOMO_MsgReturnPacket(newMsg);
epPkt = (DSR_RouteReply *) pktPtr;

epPkt->pktType = DSR_ROUTE_REPLY;
epPkt->targetAddr = rreqPkt->srcAddr;
epPkt->srcAddr = node->nodeAddr;
epPkt->hopCount = 1;
epPkt->segLeft = rreqPkt->hopCount;
for (i = 0; i < rreqPkt->hopCount - 1; i++)

    rreqPkt->path[i] = rreqPkt->path[i];           //copy the node ID's from rreq to rrep

    epPkt->path[rreqPkt->hopCount - 1] = node->nodeAddr; //enter dest ID in to last
    of array of rrep because already it has same ID just replace
    //i think every time
for (i = rreqPkt->hopCount; i < DSR_MAX_SR_LEN; i++)

    rrepPkt->path[i] = ANY_DEST;

```

```
delay = pc_eraud(node->seed) * DSR_BROADCAST_JITTER;
```

```
(rreqPkt->hopCount > 1)
```

*hop count is greater than one send it to the node
specified in rep path array rreqPkt->path[rreqPkt->hopCount - 2]*

```
if(Scout == 0)
```

```
{
```

```
NetworkIpSendRawGlomoMessageToMacLayerWithDelay(  
node, newMsg, rreqPkt->path[rreqPkt->hopCount - 2], CONTROL,  
IPPROTO_DSR, 1, DEFAULT_INTERFACE,  
rreqPkt->path[rreqPkt->hopCount - 2], delay);  
node->Ssrcaddr=rreqPkt->srcAddr;  
node->Stargetaddr=rreqPkt->targetAddr;  
node->Sseqnumber=rreqPkt->seqNumber;  
node->Shopcount=rreqPkt->hopCount;  
Scout = 1;
```

```
}
```

```
else
```

```
{
```

```
node->Ssrcaddr=rreqPkt->srcAddr && node->Shopcount < rreqPkt->hopCount
```

```
node->Sseqnumber=rreqPkt->seqNumber)
```

```
{
```

```
node->Ssrcaddr=rreqPkt->srcAddr;  
node->Stargetaddr=rreqPkt->targetAddr;  
node->Sseqnumber=rreqPkt->seqNumber;  
node->Shopcount=rreqPkt->hopCount;  
NetworkIpSendRawGlomoMessageToMacLayerWithDelay(  
node, newMsg, rreqPkt->path[rreqPkt->hopCount - 2], CONTROL,
```

```

IPPROTO_DSR, 1, DEFAULT_INTERFACE,
rrepPkt->path[rreqPkt->hopCount -2], delay);

r->stats.numReplySent++;
LOMO_MsgFree(node, msg);

RoutingDsrInitiateRREP */

led when packet is received from MAC

RoutingDsrHandleProtocolPacket(
lomoNode *node, Message *msg, NODE_ADDR srcAddr,
ODE_ADDR destAddr, int ttl)

SR_PacketType *dsrHeader =
R_PacketType*)GLOMO_MsgReturnPacket(msg);

vitch (*dsrHeader) //dsrHeader of type "DSR_PacketType
ich contain rreq,rrep,error

case DSR_ROUTE_REQUEST: //case 1: check if it is rreq?
{
RoutingDsrHandleRequest(node, msg, ttl);

break;
} /* RREQ */

case DSR_ROUTE_REPLY:
{
RoutingDsrHandleReply(node, msg, destAddr);

```

```

        break;
    } /* RREP */

case DSR_ROUTE_ERROR:
{
    RoutingDsrHandleError(node, msg, srcAddr, destAddr);

    break;
} /* RERR */
/* switch */
RoutingDsrHandleProtocolPacket */

```

**termine the routing action to take for a the given data packet
the PacketWasRouted variable to TRUE if no further handling of
s packet by IP is necessary**

```

RoutingDsrRouterFunction(
    lomoNode *node,
    lmessage *msg,
    ODE_ADDR destAddr,
    OOL *packetWasRouted)

    lomoNetworkIp* ipLayer = (GlomoNetworkIp *) node->networkData.networkVar;
    lomoRoutingDsr* dsr = (GlomoRoutingDsr *) ipLayer->routingProtocol;
    lHeaderType *ipHeader = (IpHeaderType *) msg->packet;
    lsrIpOptionType* option;
    lODE_ADDR path[DSR_MAX_SR_LEN + 1];
    llength;

```

```

l current;
(ipHeader->ip_p == IPPROTO_DSR)

return;

(ipHeader->ip_src == node->nodeAddr)

*packetWasRouted = TRUE;

se

ExtractIpSourceAndRecordedRoute(msg, path, &length, &current);
assert(length <= (DSR_MAX_SR_LEN + 1));
option = GetPtrToDsrIpOptionField(msg);
option->segmentLeft = option->segmentLeft - 1;
/* Check if received the packet before */
if (!RoutingDsrCheckDataSeen(node, path, current))
{
    *packetWasRouted = FALSE;
}
else
{
    *packetWasRouted = TRUE;
}

(ipHeader->ip_src != node->nodeAddr)

/* check if i'm the dest */
if (destAddr == node->nodeAddr && path[current - 1] == node->nodeAddr)
{
    dsr->stats.numDataReceived++;
}

```

```

/* if dest */
I'm the intended intermediate node */
se if (path[current - 1] == node->nodeAddr)

dsr->stats.numDataTxd++;
/* else if i'm the intended receiver */

ource of the route and route to the destination is known */
if (RoutingDsrCheckRouteExist(destAddr, &dsr->routeCacheTable))

outingDsrTransmitData(node, msg, destAddr);

o route to the dest is known and no Route Request has been sent */
: if (!RoutingDsrLookupBuffer(destAddr, &dsr->buffer))

outingDsrInsertBuffer(msg, destAddr, &dsr->buffer);

f (RoutingDsrLookupRequestTable(destAddr, &dsr->requestTable))
{
    RoutingDsrInitiateRREQ(node, destAddr);
}

Already sent an Route Request; just buffer the packet */
ie

RoutingDsrInsertBuffer(msg, destAddr, &dsr->buffer);

RoutingDsrRouterFunction */
ward the Route Reply

```



```
outingDsrRelayRREP(GlomoNode *node, Message *msg)
```

```
noNetworkIp* ipLayer = (GlomoNetworkIp *) node->networkData.networkVar;
noRoutingDsr* dsr = (GlomoRoutingDsr *) ipLayer->routingProtocol;
sage *newMsg;
↳_RouteReply *oldRrep;
↳_RouteReply *newRrep;
*pkPtr;
pktSize = sizeof(DSR_RouteReply);
;
↳rep = (DSR_RouteReply *) GLOMO_MsgReturnPacket(msg);
Msg = GLOMO_MsgAlloc(node, GLOMO_MAC_LAYER, 0,
_MAC_FromNetwork);
GLOMO_MsgPacketAlloc(node, newMsg, pktSize);
Ptr = (char *) GLOMO_MsgReturnPacket(newMsg);
↳Rrep = (DSR_RouteReply *) pktPtr;
↳Rrep->pktType = oldRrep->pktType;
↳Rrep->targetAddr = oldRrep->targetAddr;
↳Rrep->srcAddr = oldRrep->srcAddr;
↳Rrep->hopCount = oldRrep->hopCount + 1;
↳Rrep->segLeft = oldRrep->segLeft - 1;
(i = 0; i < DSR_MAX_SR_LEN; i++)

newRrep->path[i] = oldRrep->path[i];

(newRrep->segLeft > 1)

NetworkIpSendRawGlomoMessageToMacLayer(
node, newMsg, newRrep->path[newRrep->segLeft - 2], CONTROL,
IPPROTO_DSR, 1, DEFAULT_INTERFACE,
```

```

newRrep->path[newRrep->segLeft - 2]);

networkIpSendRawGlomoMessageToMacLayer(
node, newMsg, newRrep->targetAddr, CONTROL, IPPROTO_DSR, 1,
DEFAULT_INTERFACE, newRrep->targetAddr);

>stats.numReplySent++;
GLOMO_MsgFree(node, msg);

outingDsrRelayRREP */

node that detects the link break sends a Route Error back to the source

outingDsrInitiateRERR(GlomoNode *node, NODE_ADDR destAddr,
NODE_ADDR unreachableAddr, NODE_ADDR *errorPath)

GlomoNetworkIp* ipLayer = (GlomoNetworkIp *)node->networkData.networkVar;
GlomoRoutingDsr* dsr = (GlomoRoutingDsr *)ipLayer->routingProtocol;
GLOMO_Msg *newMsg;
DSR_RouteError *rerr;
char *pktPtr;
pktSize = sizeof(DSR_RouteError);
i;

newMsg = GLOMO_MsgAlloc(node, GLOMO_MAC_LAYER, 0,
i_MAC_FromNetwork);
GLOMO_MsgPacketAlloc(node, newMsg, pktSize);
pktPtr = (char *) GLOMO_MsgReturnPacket(newMsg);
rerr = (DSR_RouteError *) pktPtr;

```

```

>pktType = DSR_ROUTE_ERROR;
>srcAddr = node->nodeAddr;
>destAddr = destAddr;
>unreachableAddr = unreachableAddr;
>hopCount = 1;
>salvaged = FALSE;
i = 0; i < DSR_MAX_SR_LEN; i++)

rr->path[i] = errorPath[i];

workIpSendRawGlomoMessageToMacLayer(
ode, newMsg, rerr->path[0], CONTROL, IPPROTO_DSR, 1,
DEFAULT_INTERFACE, rerr->path[0]);
>stats.numErrorSent++;
outingDsrInitiateRERR */

```

ward the Route Error packet

```

RoutingDsrRelayRERR(GlomoNode *node, Message *msg)

lomoNetworkIp* ipLayer = (GlomoNetworkIp *) node->networkData.networkVar;
lomoRoutingDsr* dsr = (GlomoRoutingDsr *) ipLayer->routingProtocol;
ssage *newMsg;
;R_RouteError *oldRerr;
;R_RouteError *newRerr;
ar *pktPtr;
: pktSize = sizeof(DSR_RouteError);
:i;

dRerr = (DSR_RouteError *) GLOMO_MsgReturnPacket(msg);

```

```

Msg = GLOMO_MsgAlloc(node, GLOMO_MAC_LAYER, 0,
MAC_FromNetwork);
GLOMO_MsgPacketAlloc(node, newMsg, pktSize);
*tr = (char *) GLOMO_MsgReturnPacket(newMsg);
Rerr = (DSR_RouteError *) pktPtr;
Rerr->pktType = oldRerr->pktType;
Rerr->srcAddr = oldRerr->srcAddr;
Rerr->destAddr = oldRerr->destAddr;
Rerr->unreachableAddr = oldRerr->unreachableAddr;
Rerr->hopCount = oldRerr->hopCount + 1;
Rerr->salvaged = oldRerr->salvaged;
(i = 0; i < DSR_MAX_SR_LEN; i++)
ewRerr->path[i] = oldRerr->path[i];

workIpSendRawGlomoMessageToMacLayer(
node, newMsg, newRerr->path[newRerr->hopCount], CONTROL, IPPROTO_DSR,
DEFAULT_INTERFACE, newRerr->path[newRerr->hopCount]);
->stats.numErrorSent++;
GLOMO_MsgFree(node, msg);

RoutingDsrRelayRERR */

```

3.H file contains

```

type of packet */
typedef enum {
SR_ROUTE_REQUEST,
SR_ROUTE_REPLY,
SR_ROUTE_ERROR,
SR_PacketType;

```

```

f struct

  t_PacketType pktType;
  DE_ADDR srcAddr;
  DE_ADDR targetAddr;
  eqNumber;
  hopCount;
  DE_ADDR path[DSR_MAX_SR_LEN];
  _RouteRequest;

ef struct

  R_PacketType pktType;
  DE_ADDR targetAddr;          /* Source of the route */
  DE_ADDR srcAddr;           /* Destination of the route */
  hopCount;
  segLeft;
  DE_ADDR path[DSR_MAX_SR_LEN];
  R_RouteReply;

ef struct

  iR_PacketType pktType;
  DE_ADDR srcAddr;          /* Originator of the Route Error */
  DE_ADDR destAddr;        /* Source of the broken route */
  DE_ADDR unreachableAddr; /* Immediate downstream of broken link */
  hopCount;
  OOL salvaged;
  DE_ADDR path[DSR_MAX_SR_LEN];
  iR_RouteError;

```

```
f struct RQE
```

```
DE_ADDR destAddr;  
ktype lastRequest; /* Time when last sent a request */  
ktype backoffInterval; /* No additional Req for this time */  
t;  
t RQE *next;  
RequestTableEntry;
```

```
f struct
```

```
R_RequestTableEntry *head;  
count;  
RequestTable;
```

```
ef struct STE
```

```
DE_ADDR srcAddr;  
seqNumber;  
t STE *next;  
RequestSeenEntry;
```

```
lef struct
```

```
R_RequestSeenEntry *front;  
R_RequestSeenEntry *rear;  
count;  
RequestSeen;
```

```
def struct fifo
```

```

DDE_ADDR destAddr;
clocktype timestamp;
message *msg;
struct fifo *next;
R_BUFFER_Node;

typedef struct

DSR_BUFFER_Node *head;
int size;
R_BUFFER;
typedef struct

/* Total number of route request packets transmitted */
int numRequestSent;
/* Total number of route reply packets transmitted */
int numReplySent;
/* Total number of route error packets transmitted */
int numErrorSent;
/* Total number of data packets originated as the source */
int numDataSent;
/* Total number of data packets transmitted */
int numDataTxed;
/* Total number of data packets received as the destination */
int numDataReceived;
int numRoutes;
int numHops;
int numLinkBreaks;
int numSalvagedPackets;
int numDroppedPackets;
} DSR_Stats;

```