HISTORY BASED STABLE ROUTE PROTOCOL (HBSR)

T04429

Submitted by
M. Shams-ul-Haq
(121-CS/MS/03)
Kashif Sajjad Bhatti
(128-CS/MS/03)

Supervised By
Dr. M. Sikandar Hayat Khiyal

Department of Computer Science,
Faculty of Applied Sciences,
International Islamic University, Islamabad
(2007)

In the name of ALMIGHTY ALLAH,
The most beneficial, the most merciful

Dedicated to my parents and all my family members

M. Shams-ul-Haq

Dedicated to all those who always pray for me.

Kashif Sajjad Bhatti

FINAL APPROVAL

It is certified that we have read the thesis titled "History Based Stable Route Protocol" submitted by Mr. M. Shams-ul-Haq Reg. No 121-CS/MS/03 and Mr. Kashif Sajjad Bhatti Reg. No 128-CS/MS/03 and it is our judgment that this thesis is of sufficient standard to warrant its acceptance by the International Islamic University Islamabad for the Degree of MS in Computer Science.

COMMITTEE

External Examiner

Dr. A. Sattar
Former Director General
Pakistan Computer Bureau

[illegible]

Internal Examiner

Dr. Afaq Hussain
Head, Department of Telecommunication Engineering,
Faculty of Applied Sciences,
International Islamic University, Islamabad

[illegible]

Matta-ur-Rehman
Lecturer, Department of Computer Science,
Faculty of Applied Sciences,
International Islamic University, Islamabad

[illegible]

Supervisor

Dr. M. Sikandar Hayat Khiyal
Head, Department of Computer Science,
Faculty of Applied Sciences,
International Islamic University, Islamabad

[illegible]

A dissertation submitted to the
Department of Computer Science,
Faculty of Applied Sciences,
International Islamic University, Islamabad
as a partial fulfillment of the requirements
for the award of the degree of
MS in Computer Science

DECLARATION

We, hereby declare that this thesis, neither as a whole nor as a part thereof has been copied out from any source. It is further declare that we have developed this project and the accompanied thesis entirely on the basis of our personal efforts made under the sincere guidance of our teachers. If any part of this thesis is proved to be copied out or found to be reported, we shall stand by the consequences. No portion of the work presented in this thesis has been submitted in the support for any application for any other degree or qualification of this or any other university or institute of learning.

M. Shams-ul-Haq
Kashif Sajjad Bhatti

ACKNOWLEDGEMENT

All praise to Almighty ALLAH, the most merciful and compassionate, who enabled us to complete this research work.

We express our gratitude to our kind supervisor *Dr. M. Sikandar Hayat Khiyal* who kept our morale high by his suggestions and appreciation. He was available to us whenever and for whatever we consulted him. Without his precious guidance and help we could never be able to make such thesis.

To all our seniors, they helped us to take decisions about this thesis and provided us technical help.

And last but not least, we would like to admit that we owe all our achievements to our truly sincere and most loving parents, brothers and sisters, who mean the most to us, and whose prayers are source of determination for us.

M. Shams-ul-Haq
Kashif Sajjad Bhatti

Project in Brief

Project Title:-:

History Based Stable Route Protocol (HBSR)

Organization:-

Department of Computer Science,

Faculty of Applied Sciences,

International Islamic University, Islamabad

Undertaken By:-

1) M. Shams-ul-Haq (121-CS/MS/03)

2) Kashif Sajjad Bhatti (128-CS/MS/03)

Supervised By:-

Dr. M. Sikandar Hayat Khiyal

Head, Department of Computer Science

Faculty of Applied Sciences,

International Islamic University, Islamabad.

Tool Used:-

NS2 Simulator (Linux Based)

Start Date:-

December 2004

End Date:-

December 2006

ABSTRACT

MANETs is a network without any fixed infrastructure. Each node act as host as well as router to relay the packets destined for other nodes. Due to this characteristic of this network, the route breakage is very frequent. To solve this problem, we select the route which shows stable behavior. Our proposed solution is based on the movement history of the node. By selecting those nodes which have fair movement history, route breakage will be reduced significantly. Theoretical computation shows that routes selected on the basis of movement history are long lived

Table of Contents

List of Figures

Fig. No **Figure** **Page No**

CHAPTER#1
INTRODUCTION

1. Introduction

There are different kinds of computer networks but generally; they are divide it into two main categories.

- Wired Networks
- Wireless Networks

Wired networks are those where all sharing of data/information and resources is done on the wires connecting different systems i.e. computers, printers etc. There are different kinds of topologies used in the wired networks.

Wireless networks are those where data or information sharing is done between the systems without the presence of wires. Despite the rapid advances in technology and development of networking infrastructure, there are voids where the conventional networks do not fit in suitably, either because of economical reasons or tactical (Routing protocols designed for conventional networks are unsuitable for Wireless Networks). Such places as hill-stations and battlefields and times as natural catastrophes demand the deployment of networks rapidly and conveniently, which can form the basis of communications and data transfers.

Mobile Ad Hoc networks (MANET) are projected as a solution under such circumstances. It has been the focus of many recent research and development efforts. A Mobile Adhoc Network (MANET) is mobile wireless network composed of several mobile nodes, likely to communicate among themselves without the intervention of any centralized management or physical infrastructure.

Projected Examples of such networks includes: defense based application like War scenarios, Disaster Relief Operations like Earthquake, Rural Areas and Commercial Applications like Home networking extending internet connectivity. One of the reasons for the considerable effort in this area is that routing in such networks is a very challenging task, since network topology changes occur frequently (Routing Issues). Due to these frequent changes an established route between nodes may break. If this happens then data packets must be Re-routed quickly to make path stable. There are many other issues regarding Mobile Adhoc Networks, i.e. Security, Quality of Service, battery power constraints. But in the design of most Adhoc Network Protocols, issue of path stability (Routing) has not been addressed. Only a few protocols aim at establishing stable routes but they depend on specific hardware features or an unverified assumption. [1,5, 6, 7,8]

1.1. Basic Features of Mobile Ad-hoc Networks (MANETs) [8]

Following are the basic features of MANETs.

- There is no centralized authority for network control, routing or administration. Routing is performed by the nodes themselves.
- Frequent and dynamic topological modifications due to mobile hosts changing arbitrarily their point of connectivity. Due to movement of hosts/nodes, the neighbors of a node are constantly changing through out the network life. Neighbors of a mobile node are those nodes that can be reached by that node without the help of other nodes. So all directly connected nodes to a node are called its neighbors.
- Limited resources include energy, bandwidth, processing capacity and memory. As usually mobile nodes are either laptops or palmtops PDAs that that have limited resources in terms processor speeds, RAM and battery power.

- Network nodes play multiple roles; source/destination versus router. Nodes have to process packets even though they are not destined for that node as routing is performed through nodes themselves. This role of nodes as routers consumes processing power of nodes to process routing packets and provide relay for data packets. CPU processing needs more battery power to perform its functions. Battery power of a node is already limited when it is consumed by CPU for processing route requests or other control packets; this limitation of node becomes more severe.
- All communications, user data and control information, are carried over the wireless medium, consisting of bandwidth-constrained links. The IEEE standard for wireless communication is IEEE 802.11, MANETs also follow these standards for its specifications. IEEE 802.11 physical layer specifies bandwidths associated with sub standards. Usually INFRARED standard is used in MANETs which provide data rates 1 Mbps and 2 Mbps. Even though IEEE 802.11a, IEEE 802.11b and IEEE 802.11g provides more data rates than INFRARED. But they are still low bandwidth networks. With this low bandwidth, all communications including control packets degrades network at peak times. Low bandwidth is most harming feature of MANETs. (Stallings)
- MANETs is highly heterogeneous environment due to the diverse nature of communications technologies employed, as well as the presence of the different types of nodes. As MANETs is meant for resource sharing without using fixed infrastructure, resources may include such nodes which do not have ability to process and relay packets for other nodes e.g. printers does not have processing power to manipulate a control packet received.
- Limited survivability and vulnerability to security attacks. MANETs is still not a save way of communication of data and is susceptible to attacks which limit its survivability.

So the basic idea is to completely eliminate the concept of fixed infrastructure .

1.2. Applications of MANETs

Such networks offer unique benefits and versatility for many applications and scenarios such as disaster relief and battlefields. Projected applications of such networks include but are not limited to:

- Rescue and search operations. If in some area, natural disaster has destroyed all fixed infrastructure for communications including voice and data MANETs is a good substitute for communication between rescue/search team members.
- In Battlefield, communication between groups of soldiers can be done by using MANETs. When an army moves in a area where there is no fixed infrastructure the only option for them is use of MANETs to communicate with each other.
- A group of islands and ships communicating with the help of floating balloons and passing airplanes.
- A group of people with laptops in a conference room wants to share data form MANETs
- A group of people on a trekking trip with handsets, if demands communications then form an ad hoc network

A majority of these applications involve voice communications while some may require video transmissions (Command and Control in a war or disaster relief operation) [8].

CHAPTER#2

LITERATURE SURVEY

2.2.1. Proactive Protocols

In the proactive routing protocols group, each node maintains one or more tables containing routing information to every other node in the network. When the network topology changes, the nodes propagate update messages throughout the network in order to update their network image. A number of ad-hoc routing protocols have been proposed based on the number and structure of the routing-related tables, the network topology information exchanges and the method of maintaining a consistent network image. The Proactive protocols impose overhead to maintain the routing tables, even if many of the entries are not used at all. Their advantage is that the routes can be used at once and there is no setup delay e.g. DSDV, GSR, WRP, FSR, GPSR and LANMAR [6]

Proactive protocols are inappropriate for the ad hoc networking environment as they waste **too many wireless resources**, especially for large highly mobile networks [9]

2.2.1.1. Optimization of Original Link State Routing Algorithm (OLSR)

OLSR proposed in [10] is the optimization of original link state routing algorithm. It belongs to the class of proactive routing protocols. It is as stable as link state. Routes to all destinations are available immediately prior to the transmission. In link state all the links of as node to its neighbors are broadcasted in the entire network but it introduces the new method for flooding, multipoint relays. This can be thought of selective flooding as only elected nodes are used to diffuse the packet in the network. Moreover it also reduces the amount of packets to be flooded in the network by sending information of only selected nodes called multipoint relay selectors.

We discuss OLSR in detail as it is candidate of proactive routing.

Multipoint relays

Multipoint relays are used to minimize the flooding of broadcast packets. Each node selects a set of its one hop neighbors to broadcast the packet to avoid the retransmission of the packets in the particular area. This set of neighbors who is responsible for broadcasting the packet is called multipoint relays (MPRs) of that node.

All the neighbors of a node read and process the packet but do not retransmit the packet. For that reason every node maintains list of its neighbors from whom if it receives broadcast packet will also retransmit the packet. This list of neighbors is called MPR selectors each node select its MPRs in such a way that when they transmit packet it must reach all two hop neighbors. Figure shows the multipoint relay selection for node N (figure borrowed from [10])

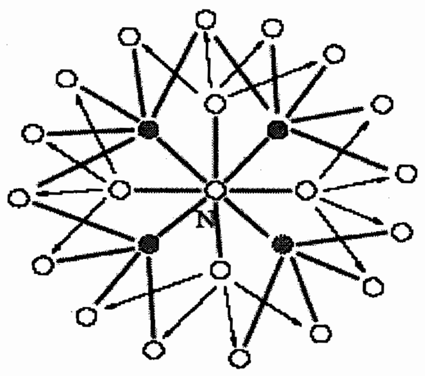

Figure 2.1: Multipoint Relay Selection for node N

OLSR selects these MPRs as intermediate nodes for a path to each destination in the network. To accomplish this task, each node broadcast information about nodes who have selected it as their multipoint relays. So a route from a source to destination is sequence of

MPRs. As MPRs are selected on bi-directional link so acknowledgement lost problems occurred due to unidirectional nature of a link are automatically avoided.

Protocol functioning

The functionality of OLSR consists of 4 modules

i) Neighbor sensing

ii) Multipoint relay selection

iii) MPR information declaration

iv) Routing table calculation

i) Neighbor sensing

Each node must have knowledge of its neighbors up to two hops in order to select its MPRs. To get knowledge of its neighbors, a node periodically broadcast HELLO packet which is transmitted to all neighbors but is not propagated further. A HELLO packet contains addresses of neighbors to which it has bi-directional link and a list of addresses which this node has heard (still not decided as bi-directional links); if node finds its own address in this list it label it as is bi-directional link. HELLO packet also contains a list of MPRs due this list each node can construct its MPRs selector table.

After exchange of this information, the neighbor table of each node contains information about its one hop neighbors, link status (bidirectional, uni-directional, MPR), and list of two hop neighbors that can be reached through one hop neighbors. Each entry has a timer in order to remove old entries and a sequence no to maintain a fresh entry.

ii) Multipoint relay selection

MPR set is calculated in one hop neighbors in order to reach two hop neighbors. This set is recalculated when a new bidirectional link is up or when a bidirectional link goes down either in one hop or two hop neighbors. MPR selector table for a node consists of those one hop neighbors that have selected that node as their MPR so this table for a node can be constructed from HELLO packet information as it contains a list of MPRs of that node which have sent this hello packet. A sequence no is also attached with this table which is incremented each time this table is updated.

iii) MPR information declaration

Each node in the network maintains another table topology table which is constructed from information received from other nodes of the network through specialized packets called topology control(TC). Each node periodically sends a TC message in the network to declare its MPRs selectors. This information is used by other node to construct a neighbor table. A node which has empty MPRs selector list will not send TC message.

On receipt of TC message topology table is constructed as

If there is entry for originator of the TC message and MPR selector sequence no is greater than that in message then it will simply discarded considering it delivery out of order. But if incoming MPRs selector sequence no is greater then topology entry is removed. If there is entry in the table that's destination is the MPRs selector and last hop is the originator then entry will be refreshed otherwise new entry will be added in the table.

iv) Routing table calculation

Each node maintains a routing table containing a entry to reach each possible destination in the network. Each entry in the table consists of destination address, next hop address and estimated distance to the destination. Entries for partially known routes or broken routes are not recorded. The routing table is built by tracking the last hop-node pairs in the descending order. To find a path given source to remote node R, one has to find connected pair (X, R) then a connected pair (Y, X) and so on till one find Y in the neighbor list of origin. Figure explains this procedure.(original figure from [10])

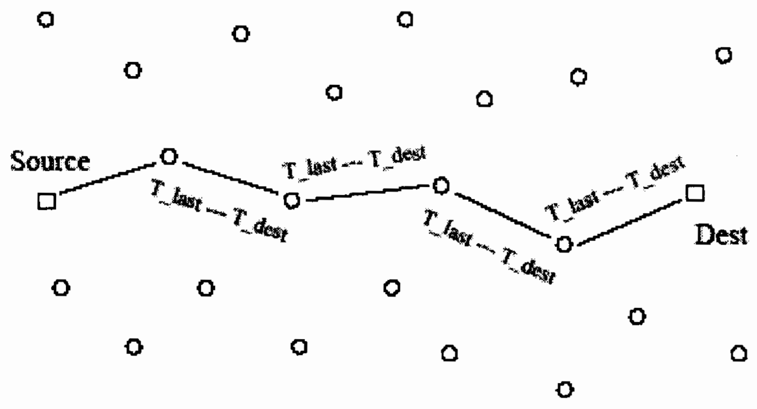

Figure 2.2: Routing Table Calculation [10]

For routing table calculation first all entries in the table are deleted. For all one hop neighbors (h=1) with bidirectional links, an entry is made in the table. Next for two hop neighbors (h=2) procedure is as

If against an entry in topology table there is no entry in routing table and last-hop address corresponds to one hop destination then a new entry in the routing table is added. In this entry destination address presents address in topology table and distance is set to h+1. In this way a route entry for all possible destinations in the network is added.

Now we see the packet format used in OLSR as defined in OLSR rfc3626

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Packet Length         |    Packet Sequence Number     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Message Type |     Vtime     |         Message Size          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Originator Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time To Live |   Hop Count   |    Message Sequence Number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                            MESSAGE                            :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Message Type |     Vtime     |         Message Size          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Originator Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time To Live |   Hop Count   |    Message Sequence Number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                            MESSAGE                            :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                                                               :
         (etc.)
```

Figure 2.3: OLSR Packet Format[11]

As OLSR is proactive in nature it is well suited where routes to new destination are very frequent. OLSR is well suited where application demands no delay in transmission but in case where data transmission in the network is less then its performance degrades. Maintaining four tables on each node is very expensive in terms of network bandwidth.

OLSR degrades much when there is large movement of nodes in the network. That's why this protocol is considered very expensive

2.2.2. On-Demand or Reactive Protocols

In contrast to the table-driven proactive routing protocols that keep concise routes or paths towards every node, reactive routing protocols calculate routing information only when data are ready to be transmitted adopting a lazy routing approach. The calculated path is considered valid as long as the destination is reachable or until the route is no longer needed.

Reactive routing protocols avoid the traffic flood and the overhead of periodic routing calculation, but they insert delays at routing time. Thus, they best fit on large, rapid changing environments, like sensor networks. [22]

The advantage of on-demand routing protocols lies in the fact that the wireless channel (a scarce resource) does not need to carry many routing overhead data for routes, which are not even used. This advantage may diminish in certain scenarios where there is a lot of traffic to a large variety of nodes. Thus, the scenario will have a very significant impact on the performance. In such a scenario with lots of traffic to many nodes, the route-setup traffic can grow larger than the constant background traffic to maintain correct routing information on each node. Still, if enough capacities would be available, the reduced efficiency (increased overhead) might not affect other performance measures, like throughput or latency.[4]

We also consider some location based protocols as on-demand protocols, since they determine the direction in which to send the packet on demand and some protocols may even initiate a location query of the destination nodes for their packets on demand.

Examples for on-demand protocols are the following:

ABR, AODV, CEDAR, DREAM, DSR, FORP, GEDIR, LAR, SSR, WAR [6]

2.2.2.1. Ad hoc On Demand Distance Vector (AODV)

As an example of on-demand routing protocol we discuss here AODV which is based on Distance Vector routing strategy of wired networks The Ad hoc On Demand Distance Vector (AODV) routing algorithm is a routing protocol designed for ad hoc mobile networks. AODV is capable of both unicast and multicast routing. It is an on demand algorithm, meaning that it builds routes between nodes only as desired by source nodes. It maintains these routes as long as they are needed by the sources. Additionally, AODV forms trees which connect multicast group members. The trees are composed of the group members and the nodes needed to connect the members. AODV uses sequence numbers to ensure the freshness of routes. It is loop-free, self-starting, and scales to large numbers of mobile nodes.

The design is based on DSDV (Distance Sequenced Distance Vector) routing algorithm with the aim of reducing the need for system wide broadcasts as much as possible. With AODV small changes in network topology due to mobility does not require system wide broadcasts. This localizes the effects of local movements whereas in DSDV local movements have global effects. Broadcasts in DSDV have been replaced by careful bookkeeping that identifies if one or more links are broken. Whenever routes are not used they are expired and consequently discarded, which reduces the effects of stale routes as well as the need for route maintenance for unused routes. Multiple routes can be used when they are collected during the discovery phase. However there are several potential difficulties with this approach: complexity in managing the ageing process for several routes; complexity involved when an alternate route goes stale and complexity in book keeping when two alternate routes share a common link which eventually breaks!

Hence current implementations of AODV use a single route for unicast destinations.

Working

AODV stands for Ad-hoc On-demand Distance Vector routing and is, as it states, an on demand protocol. It provides loop-free routes through the use of sequence numbers associated to each route. In short, if A needs a route to B it broadcasts a ROUTE REQUEST message. Each node that receives this message, and does not have a route to B, rebroadcasts it. The node also keeps track of the number of hops the message has made, as well as remembering who sent it the broadcast. If a node has the route to B it replies by unicasting a ROUTE REPLY back to the node it received the request from. The reply is then forwarded back to A by unicasting it to the next hop towards A. This establishes a uni-directional route (asymmetrical link). For a bi-directional route (symmetrical link) this procedure will need to be repeated in the reverse direction. To achieve faster convergence in the net, and thus higher mobility, a ROUTE ERROR message can be broadcasted on to the net in the case of a link breakage. Hosts that receive the error message remove the route and re-broadcast the error messages to all nodes with information added about new unreachable destinations.

A More Detailed View:

AODV builds routes using a route request / route reply query cycle. When a source node desires a route to a destination for which it does not already have a route, it broadcasts a route request (RREQ) packet across the network.

Destination IP Addr.	Dest. Sequence Nr.	Hop Count	Next Hop	Precursors	Lifetime	Flags
..	..	..	..	..	..	..

Figure 2.4: Packet Structure

Nodes receiving this packet update their information for the source node and set up backwards pointers to the source node in the route tables. 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. Nodes keep track of the RREQ's source IP address and broadcast ID. If they receive a RREQ which 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.

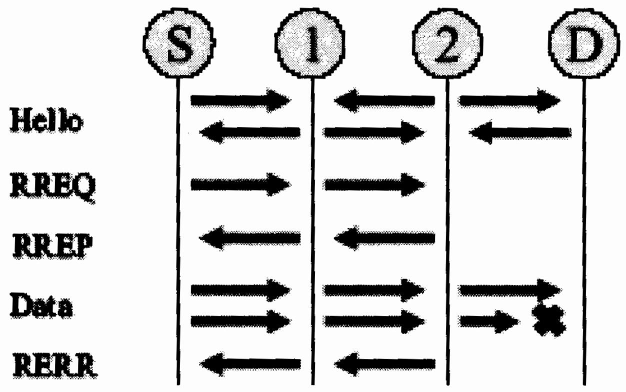

Figure 2.5: AODV Protocol Messaging[16]

As long as the route remains active, it will continue to be maintained. A route is considered active as long as there are data packets periodically traveling from the source to the destination along that path. Once the source stops sending data packets, the links will time out and eventually be deleted from the intermediate node routing tables. If a link break occurs while the route is active, the node upstream of the break propagates a route error (RERR) message to the source node to inform it of the now unreachable destination(s). After receiving the RERR, if the source node still desires the route, it can reinitiate route discovery.

2.2.3. Hybrid Protocols

Also, there are protocols (as to say protocol sets) that utilize both proactive and on-demand routing. The examples are ADV and ZRP

ADV - Adaptive Distance Vector routing routes are maintained proactively, but only to certain nodes (active receivers), and the size and frequency of the updates is adapted. So the authors claim it is a hybrid protocol.[4]

ZRP - Zone Routing Protocol also consists of a proactive Intra Zone Routing Protocol (IARP) and an on-demand Inter Zone Routing Protocol (IERP).[4]

2.3. Security in MANETs

Securing the mobile ad-hoc network is still a great issue and very little work has been done in this case. All the network security threats that can harm the normal wired networks are also equally damaging for the MANETs [7]. These security threats include DOS attacks, sniffing, spoofing etc. In MANETs DOS attacks can easily launched against a particular node where that node forced to re-initiates the discovery process repeatedly. Packet sniffing is also a key issue while securing the MANETs. As all the packets are broadcasted in whole network, therefore packet sniffing is very easy. Similarly, a group of malicious nodes can launch a *Black Hole* attack, which is a kind of IP spoofing. No single routing protocol deals with these threats. Therefore, MANETs are very vulnerable to security threats [12].

Some extensions in AODV are proposed to tackle some of the security threats like "Black Hole Attack".

As AODV is on demand protocol, it discovers the different paths to destination when it is required. In " Black Hole " attack, a malicious node uses the routing protocol to advertise itself as having the shortest path to the node whose packets it wants to intercept [13].

In normal AODV for route discovery request packets are flooded to nearby nodes and if the neighbor node is destination or an intermediate node has the fresh enough path, it will reply to its previous node and so on. While route discovery process, the first reply is selected as a path and rest are discarded.

So, Black Hole attack can easily be launched in such simple case. e.g. if a nearby node is malicious, when the request packet comes to it, it just send the reply message as if it has the fresh path available to destination. As only the path sending the first reply message is selected and rest are discarded, so in above case the malicious node can send the reply message very soon as it does not need to check its routing table. Therefore, when the source node selects the path, it starts transmitting the packets and the malicious node can get all the information.

Therefore, the proposed system is to have some extra control messages. When an intermediate node send the reply messages to source node this message also has the next hop address then the source node send the further reply message to next hop and ask it whether it has route to destination and has a path to the node sending the reply messages. The answer to this message can only be considered correct by the next hop node. If the malicious node tries to send the message, it simply discarded. When the next hop reply to message it contain all the required information as it has route to destination and to the node sending the reply message. If the answer is yes then the source node starts sending the data packets and if the answer is no it will broadcast the alarm message to whole network and makes this malicious node isolated.

As nothing is free so this also cost in terms of battery power and bandwidth, so in normal networks, we don't use it all the time. Whenever we feel any uncertain activity, we activate this process to detect and locate the malicious node.

2.4. Quality of Service Routing (QoS)

Quality of service (QoS) can be defined as

Set of guaranteed parameters defining behavior of the network under certain conditions [7]

In war and disaster relief scenarios, some connections may be more important than others. A message of extreme urgency would want to reach the destination without any network delay or disruption. The current TCP/IP model of the Internet, which can only support best-effort service, cannot satisfy these requirements. Although TCP assures that the packets reach their destination safely, it does not guarantee any bound on the delay, nor can it assure a minimum bandwidth for the connection. Thus, there is a need for a QoS architecture that can provide the required service differentiation as well as deterministic service quality to the demanding connections.

An ideal QoS model should be able to differentiate flows based on priorities and provide deterministic service guarantees to the admitted flows while making a good utilization of the network resources. Many QoS models have been proposed for the conventional networks.

However these models are not directly applicable to the paradigm of ad-hoc networks due to the characteristics of these networks such as mobile nature of nodes resulting in unpredictable topology, scarce wireless bandwidth which varies at the mercy of environmental conditions, limited power of the nodes and peer-to-peer nature of nodes requiring co-operation for relaying other's packets among many. These characteristics not only make ad hoc networks very different from the conventional Internet, but also make the task of providing QoS assurances in these networks extremely challenging [14].

2.5. Parameters of the QoS

There are certain QoS parameters depending upon the application type [7]. These are

1. ***Multimedia applications***
    - Guaranteed bandwidth
    - Delay (not higher than ...) & delay jitter
2. **Search-and-rescue operations**
    - Accessibility (availability) to the network
3. **Military networking**
    - Security requirements
4. **Sensing networks**
    - Power consumption efficiency

2.6. QoS Challenges in MANETs

Ad hoc networks differ radically from the conventional wired or cellular networks. The characteristics of these networks make the task of providing QoS guarantees in these networks extremely difficult [7]. Some of these difficulties are well known and well pointed out in many papers. These challenges are discussed briefly below .

2.6.1. Unpredictable Network Topology

An ad hoc network consists of nodes that can move arbitrarily in random directions and with different speeds. Further, these nodes are not affiliated to any fixed base station. The network formed is thus dynamic and has an unpredictable, time varying topology. The rate of change of topology depends upon the speed and the movement patterns of the nodes. For example, a collection of ships moving in the same direction may have a stable topology even if they are moving fast. The constant change in topology results in inaccurate state information at the nodes making convergence of routing information difficult. The capacity of the network and the utilization of its resources changes with the speed of the nodes. This implies that if the parameters of a QoS model within the nodes of the network are tuned to operate at a particular speed, the model might fail to provide assurances if the nodes start to move faster. Either the parameters must be re-tuned, or the nodes must be made aware of the average speed of movement of the network nodes, so that the parameters can be adjusted dynamically. It was also seen that at very high speeds, the network capacity degrades exponentially due to excessive link breaks. In order to make any assurances, it is important that the rate of change of topology is not ”too” fast; otherwise, flooding may be the only routing solution.

2.6.2. Scarce Radio Bandwidth

Scarcity of radio spectrum implies that only a limited amount of bandwidth will be available to a MANET node. This bandwidth is further influenced by external conditions such as weather, physical obstacles, fading, noise and interference from outside sources and from other distant nodes [7].

A simple scenario where these mechanisms do not work is when two nodes are within interference range of each other but not in the transmission range. Such a scenario is shown in Figure 2.6.

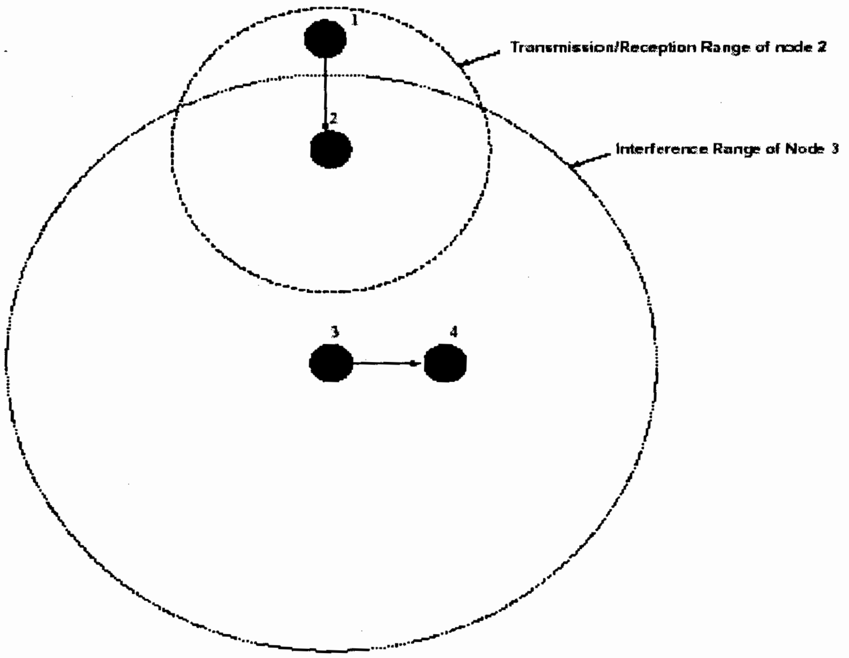
1
2
3
4
Transmission/Reception Range of node 2
Interference Range of Node 3

Figure 2.6: Interfering Nodes [10]

As shown, node 1 has data to transmit to node 2 and node 3 has data to send to node 4.

When node 1 wants to send a data packet, it waits for the medium to be idle for a specified period. It then transmits a RTS (Request-to-Send) packet. Since node 1 is outside the interference range of node 3, it always senses the medium idle. If node 3 transmits a packet to node 4 while node 1 transmits to 2, a collision takes place at node 2. Then, node 2 does not receive the RTS packet correctly. The collision can also take place for a data packet. If, upon receiving the RTS packet, node 2 senses the medium idle, it replies with a CTS (Clear-to-Send) packet. This packet informs the neighboring nodes about the forthcoming transmission of data packet and hence effectively reserves the medium for the duration of transmission of the packet. However, node 3 is outside the reception range of node 2, hence it does not receive the CTS packet. Node 1 receives this packet and assumes that the medium has been reserved. It then starts its data packet transmission. If node 3 now starts transmitting a packet to node 4 before sensing the medium to be busy, a collision takes place at node 2, resulting in loss of data packet. These collisions reduce the effective bandwidth at node 2. In such a situation, node 2 has no means of reserving the medium for transmission or reception. In that case, it cannot accurately estimate the bandwidth availability for making long-term reservations. With the unavailability of accurate information, mechanisms such as admission control and rate control for the maintenance of admitted flows is very difficult. Further, the limitation of available bandwidth requires that the exchange of control messages be limited. This restriction may require that nodes transmit control updates less frequently or restrict them to immediate neighbors. This results in nodes having stale information of the network conditions until the next update and further delays the convergence of control information.

2.6.3. Limited Battery Power

The mobility of nodes limits their size, which in turn limits the energy reserves available to them. Thus, energy conservation is a key requirement in the design of ad hoc networks .

Transmission of packets is the single largest consumer of power. Transmission power control used for communications impacts the operational lifetime of devices in different ways. For devices, where the transmission power accounts only for a small percentage of the overall power consumed, (e.g., a wireless LAB radio attached to a notebook computer) reducing the transmission power may not significantly impact the device's operational lifetime. In contrast, for small computing/communication devices with build-in or attached radios (e.g. cellular phones, PDAs, sensors, etc.) reducing the transmission power may significantly extend the operational lifetime of a device, thus, enhancing the overall user experience.

Hence, for such devices, it is necessary that the numbers of control packets transmitted be minimized. For a QoS framework, it is necessary to exchange control information related to bandwidth availability in the neighborhood, admission control and quality feedback. A reduction in the transmission of such packets leads to the problems related to inaccurate state information and may cause flows to lose their quality. Designing a protocol with contradicting goals of achieving desired levels of QoS while reducing power consumption due to transmissions is a challenging task [19].

2.6.4. Path/Link Stability

There are many proposed QoS models but the QoS metrics for all the models have one thing in common and that is path stability. Nothing can be achieved either its battery power constrained or delay or jitter constrained without the stability of paths. Therefore, QoS routing is an essential part of QoS architecture. It is a routing mechanism under which paths for flows are determined on the basis of some knowledge of the resources available in the network as well as on the QoS requirements of the flows or connections.

The most common routing protocols like AODV, DSDV, DSR and TORA are not guaranteed for the stable paths [21]. Therefore, many extensions in already existing protocols and some new protocols are introduced to have stable paths in MANETs. Having stable paths or long-lived paths in mobile ad-hoc networks is extremely difficult because of frequent node mobility and unpredictable network topology. If the node mobility is very rapid and nodes are moving at a great speed then most of the techniques to have stable paths will not work, therefore there should be some sort of controlled mobility of nodes, to have stable paths.

2.7. Strategies of QoS routing protocols

We can classify all the protocols providing path stability into two categories.

- Improving the route repair procedure.
- Selecting the route whose nodes have most stable behavior.

2.7.1. Improving the route repair procedure

Those who favor the reactive protocols because of its less consumption of bandwidth and network resources provide some extensions to get stable paths in MANETs [12].

Therefore, the focus in such techniques is not on selecting the nodes having stable behavior but to improve the link repair procedure. This includes the fast detection of link breakage and fast repair procedure to reduce the delay and bandwidth consumption. These protocols mainly use the following procedures.

Route repair

If a failure occurs during communication between two nodes, two scenarios can be used to repair the route.

2.7.1.1. Local route repair [1]

When a node detects a link failure, it does not systematically send an error message to the source. First, it attempts to repair the route itself. It only sends an error message to the source if this first attempt fails. Consider the case presented in Figure 2.7. Step (a) represents the route before failure detection. In step (b), Node D detects a link failure between D and G. According to the local route repair procedure of step (c), D diffuses a route request (RREQ) packet, which is propagated across the network. When it receives this packet, J returns a route reply (RREP) packet to D. This packet is forwarded by I, G and F.

When D finally receives it, the route is repaired and communication can carry on.

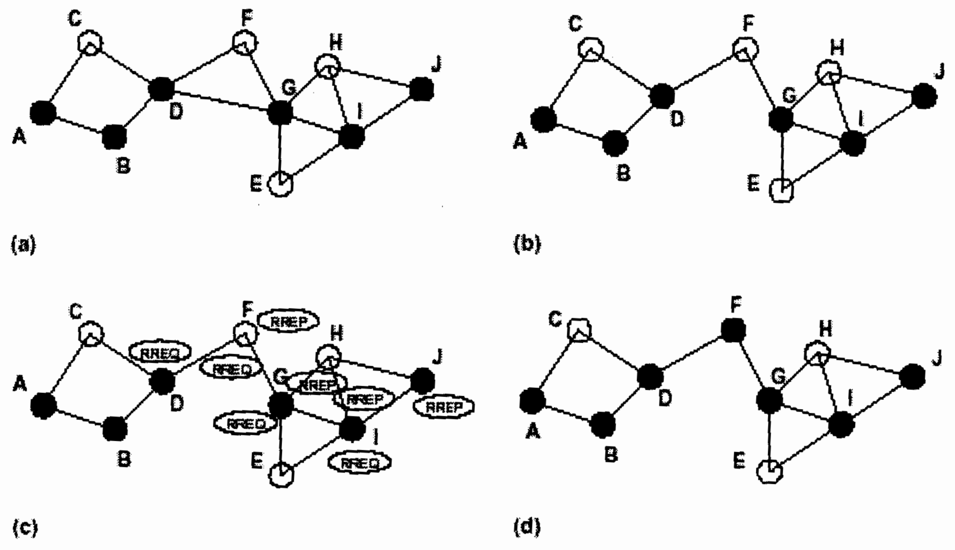

Figure 2.7: Local route repair: (a) initial route, (b) ruptured initial route, (c) local repair procedure and (d) new repaired route. [1]

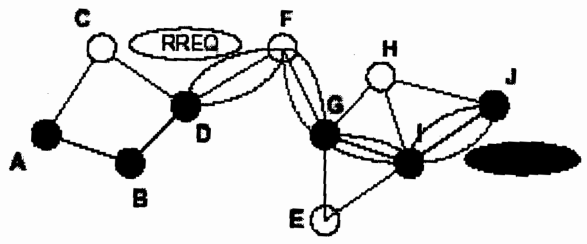

Figure 2.8: Local Route Repair. [1]

2.7.1.2. Global route repair [1]

If a link failure were detected during the transmission of a packet from source A to destination B, the node that detected the failure would return an error message to the source. Then, the source initiates a new route discovery phase to find a new path between A and B. This phase requires much time to complete and overloads the network with many routing messages. This results in bandwidth waste that is detrimental to the overall performance of the network.

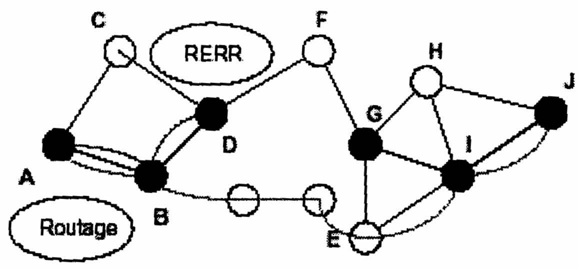

Figure 2.9: Global Route Repair. [1]

The primary objective of these protocols is to insure the selection of the most easily reparable route among those extracted from the route discovery phase. To achieve this goal, these protocols recommend taking into account the nature of the neighborhood of the nodes composing the network, and in particular the density of nodes and their availability. The reparation of a route in the case of failure can be carried out through local route repair that is least expensive than the global repair or the normal repair strategies .

The availability parameter is used to establish the ability of Node A to replace Node B. The availability of a node depends on the nature of the node (laptop, PDA, etc.), the number of packets that are forwarded by the node on its communication channels as well as their capacity. In short, the availability corresponds to the available bandwidth of a node for each of its communication channels. Since each node needs to be aware of the availability of all of the other nodes in its neighborhood, this can be done by providing the periodic availability information exchanges within the nodes of a neighborhood.

We define the density of a node λ as the number of direct neighbors of λ (that is, the number of nodes in the k radio range) whose available bandwidth is higher than that required by the connection. The density parameter is completely specified by a node and the bandwidth associated with that node.

The average density can be calculated using the given formula

$$D_m = \frac{D_m(n-1)+D}{n},$$

The discovery phase:

We use the route discovery phase as in the AODV protocol for which we add provisions for the availability and density parameters. These two parameters need to be taken into account in order to provide QoS. Thus, the source initiates the routing process upon receiving a connection request. Then, it sends a route request for this connection to all of its neighbors. The nodes that receive the message for the first time and that fulfill the QoS requirements propagates the request message towards the destination.

The request message is gradually propagated towards the destination following the fore-mentioned scenario. Finally, when it arrives at its destination, the destination node initiates a countdown and records all of the incoming message requests. At the end of the countdown, the destination selects the easiest route to repair and then sends a confirmation packet to reserve the needed resources.

To select a route, the parameters contained in each message request that arrived at the destination are necessary. It is important to note that a route containing long sequences of high- density nodes will be easier to repair with the local route repair procedure than a route that does not hold that property. The repair fitness F associated with a given route is expressed as

$$F = \frac{D_m}{n+1+e(5/2-1/2n)},$$

Where e is the number of strangulations on the route, we select the route whose repair fitness is the highest among all routes. So the final route that is selected using the repair fitness formula has more chances to use local repair procedure in case link failure.

Such routing methods achieve QoS routing by consuming less bandwidth than the pro-active protocols The main focus is on the easy repair or local repair of routes based on node density not on the node behavior. There also come such situations where an attempt to repair a route locally could be useless and even harmful for the network: if the density around the node, which initiated the local route repair, is too low, the route repair procedure is bound to fail. In this situation, it is preferable to turn over directly to the source after having detected the failure of the link and to proceed to a global route repair.

Cluster Head Selection

Some other protocols such as [4]use the hybrid approach to repair the broken link or paths efficiently. In such techniques the network is divided into several non-overlapping clusters and adopts proactive routing protocol inside the clusters and reactive routing protocol outside the clusters. When two mobile hosts communicate to each other, the routing path connecting these two mobile nodes may be through several intermediate clusters. Therefore, the maintenance efforts of the entire routing path can be distributed into several intermediate clusters. [4]

TH-4429

Nodes that have "local maximum degree" are selected to cover the dense area forming stable clusters. Besides, the cluster structure is designed as a gravitational area to reduce broadcasting control packets for cluster maintenance. Each node of clusters knows its potential of the gravitational area, and only the cluster head knows the entire cluster topology. The active connection can be repair efficiently and locally by light maintenance cost and stable clusters. In order to achieve global repair efficiently, GCR (Gravitational

Cluster Routing) quantifies the repairable level of a route and use it to find routes that have stronger connectivity between clusters to overcome the high mobility of clusters.

When the cluster id of a node and all its neighboring nodes are null over a particular periodic time, the node broadcasts degree information to its all neighbors. The node that is not a member of any cluster and receives the degree information from its neighbor will broadcast its degree information to its all neighbors again. After this procedure, the node whose degree is greater than its all neighbors will be a cluster head.

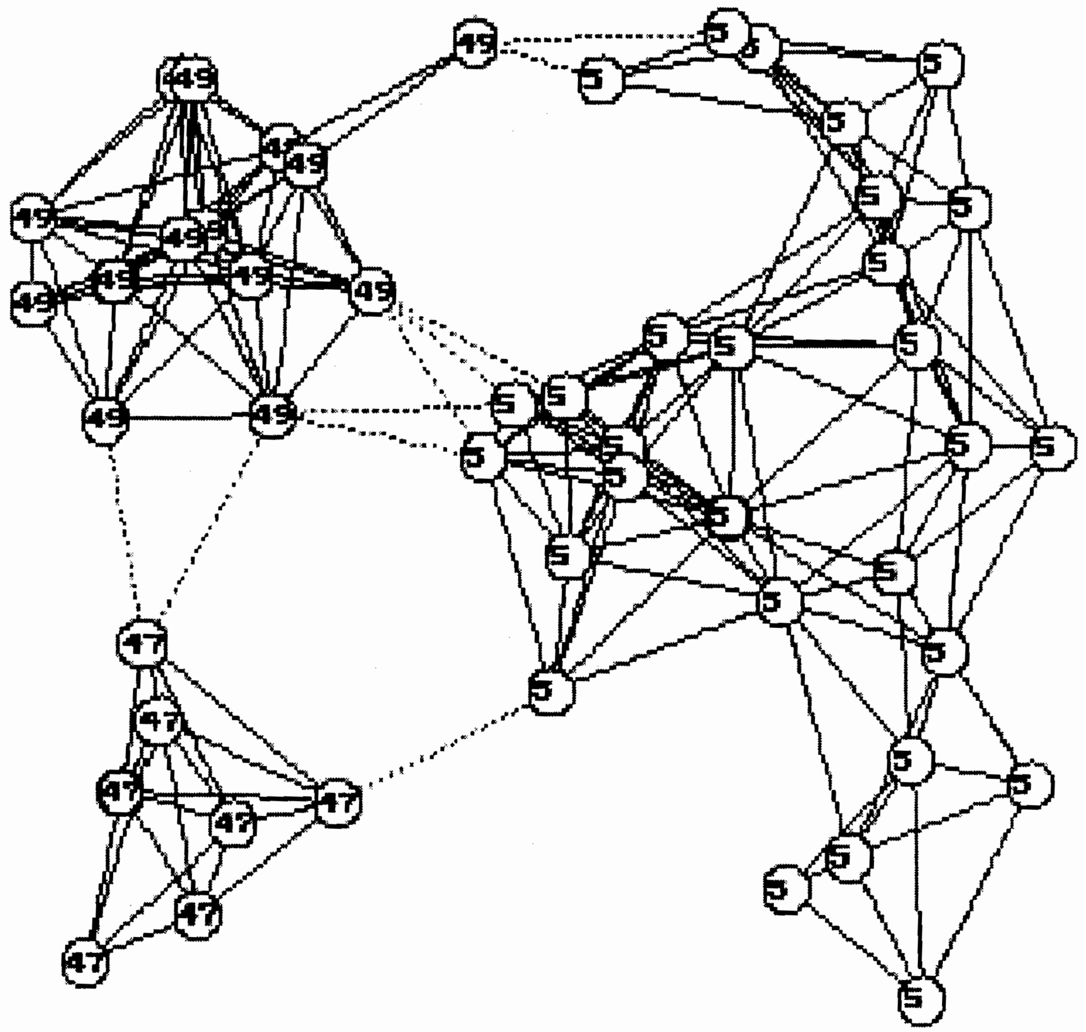

Figure 2.10: Showing the different clusters and cluster heads [4]

After the cluster head selection the cluster reconstruction starts and the cluster head broadcast the control packets to all the nodes and then forming the cluster. All the nodes in the cluster is maintained pro-actively and the routing id done between the clusters reactively.

If link breakage occurs inside the cluster, then such node will put outside the cluster, it will wait for some time to renter in some other cluster, and if the link breakage occurs between the inter-cluster nodes, then the cluster head will select another node to re-construct the path.

These all protocols work best in such networks where the node density is high and the mobility of nodes is low.

2.7.2. Selecting the route whose nodes have most stable behavior

In this approach, routing is done pro-actively and those routes are selected that have stable nodes behavior. The information about these nodes is kept in the routing table and some other tables depending upon the technique implementation. As in pro-active routing, the consumption of bandwidth to maintain the routing tables is greater than the reactive routing where path discovery is done on-demand basis. These techniques are implemented in order to minimize the probability of link breakage by spending the continuous bandwidth because the discovery process is very expensive and consumes a lot of network resources i.e. network bandwidth, battery power and also produces delay in the communication. Many different protocols evolved with the time for this purpose.

"*An Adaptive QoS Routing Protocol with Dispersity for Ad-hoc Networks*" is the one of the protocol that uses the node's signal strength to have stable paths [3].

In this paper, the authors propose a new route discovery algorithm to find multiple disjoint paths with longer lived connections, where each path also specifies associated network information. As long-lived path, generally less maintenance results in less bandwidth utilization. Routing is pro-actively done before path unavailability occur.

As there are multiple paths exists so switching from one path to another reduces delay significantly. Signal strength is used for route maintenance. Multiple paths are used hence less congestion and efficiently utilization of bandwidth.

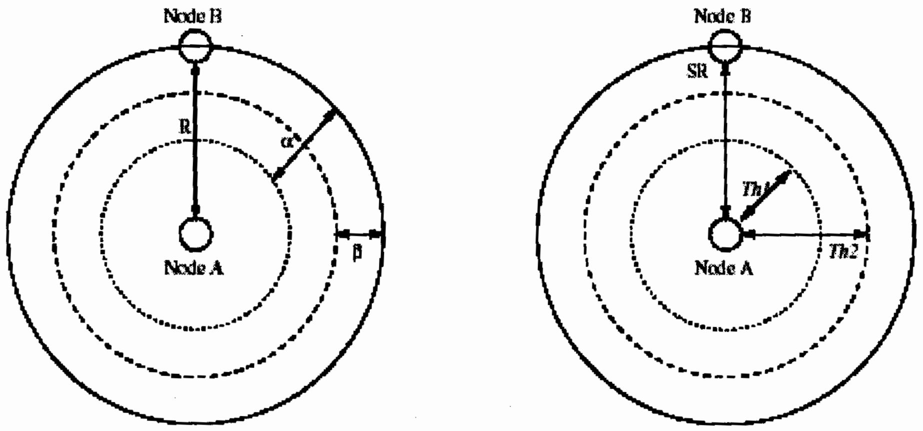

Figure 2.11: Two thresholds [3]

They classify the nodes, links and routes in to three classes

1. A node whose signal strength is greater than certain threshold1 is first class node.
2. A node whose signal strength is between threshold1 and threshold2 is second-class node.
3. A node whose signal strength is less than threshold 2 but greater than SR (minimum signal strength where communication takes place.) is third class node.

Similarly, the links classes, as link of a node with first class node are first class link and go on.

Also route classes, a route containing first class links is first class route, a route containing second class links is second-class route and similarly third class route.

Each node maintains **network table,** which updated lists of its neighbor, updated signal strength and minimum signal strength to transfer data.

In **NT** "+" means neighbor is moving away and "-" means neighbor is moving closer.

Each node also creates a **Routing table**, which has updated lists of all the possible routes to desired destination. Each element in **Routing table** is eleven tuple.

<Source, destination, next hop, hop count, available bandwidth, reserved bandwidth, active, route class, first class link, second-class link, third class link>.

Now for route discovery the source broadcast the **route_request** packet that has

<source, destination, request id, hop cnt, QoS metric ,route class, int nodes, first class link, second class link ,third class link>

If reply is not come in given time threshold, the packet is retransmitted and if extra packets arrived then they are discarded. The paths are selected such that they are disjoint and if a node receives a packet for the same destination again, it discards the packet.

A QoS_Reserve or Qos_Release packet is used to reserve and release the bandwidth from source to destination node.

There are two phases of monitoring the re-routing

- Pre-Routing
- Re-Routing

In pre-routing basic operation are performed to route and in re-routing actual routing is done

As request packet arrives at destination from different paths, it check the bandwidth availability and link class and send request reply packet to destination over the same paths. The source then pick up the best path depends upon the bandwidth available and link class and starts data transmission. And all the second best paths reserves for future uses now the it constantly keep tracking of its neighbor, if a node move away or move closer to it, it keeps updating its **RT** table.

Now if any node in path is moving away from its neighbor it starts looking for another best alternate path, which it already have without disrupting the communication.

As the path becomes unavailable or going to be unavailable, it switches to another path immediately. Moreover, if it does not have any path, it starts the process of path discovery while the data is still transmitting and reserve the bandwidth and then switch to it and releases the previous bandwidth and updating the **RT** table. But while rediscovering of the new path the request packets are not sent to the weak nodes as they are not stable or going to become unavailable in near future. So, by doing this QoS routing achieves, as QoS is the metrics of efficient bandwidth utilization and less delay.

While in another technique to have a stable path in the MANETs, the basic focus is on the long life of the path rather than the optimal path. This protocol proposed an approach involving reliability based query restriction [15]. This approach also helps in reducing the routing overhead i.e. flooding while discovering the path and hence reduction in congestion. This protocol estimates the life of the link from the past-perceived behavior of the link. This approach may select a long path if it is long lived rather than selecting an optimal short-lived path. The reliability of the node is continuously evaluated and updated. They define a certain threshold value for reliability, if reliability exceeds that threshold value than link is good. As

Ad-hoc protocols generally tend to have, the neighborhood management stability built into it. This paper uses the same feature of Ad-hoc protocols.

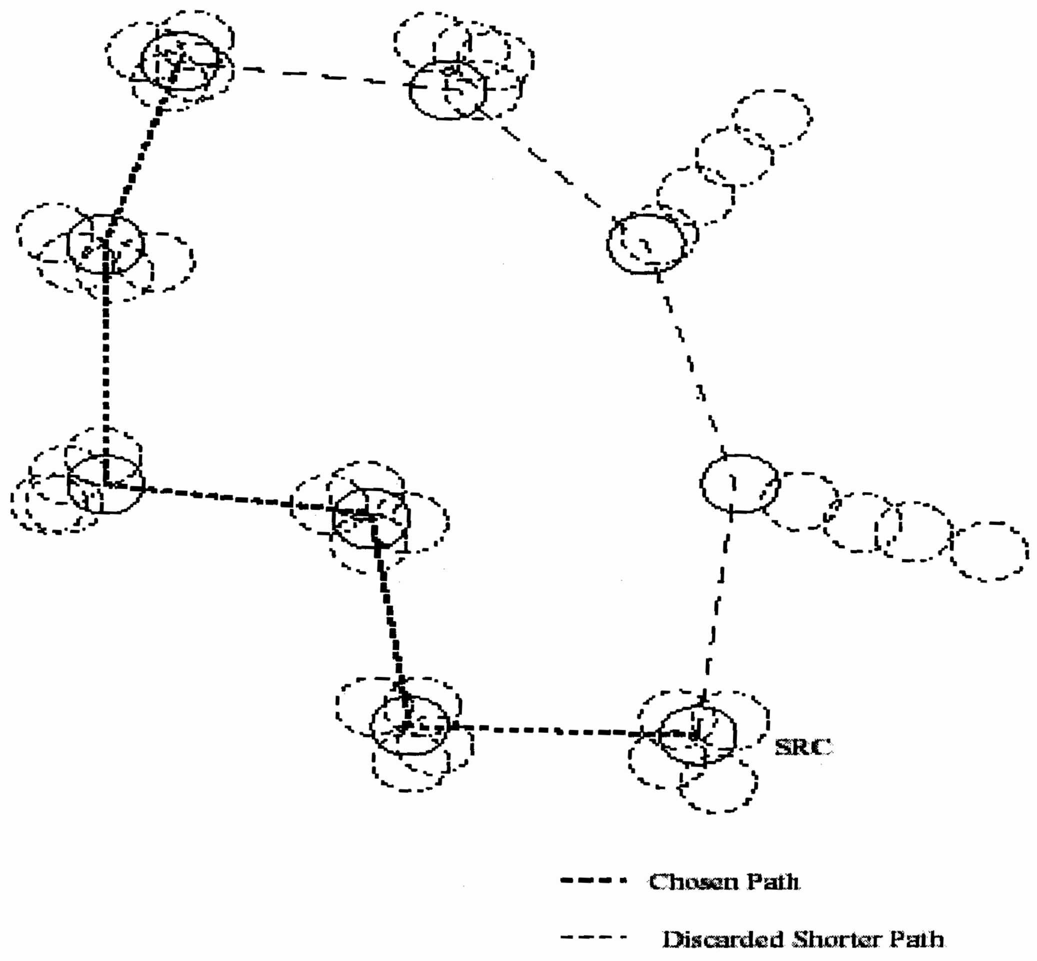

Figure2.12: Using a reliable route while ignoring shorter but unreliable ones [15]

Therefore, the focus is on the presence of nodes in the immediate vicinity of a certain node. Every node maintains a reliability value of every other node that remains his neighbor. They check it by sending the Hello messages. Whenever a node receives the Hello message, it will increment its reliability value by some factor. When the link breaks or it doest not receive the message it will decrement the reliability value. Whenever a node is added to the neighbor-list, its reliability is set to link life time and its value gets decremented by 1 every second. Similarly, the congestion will be reduced by sending the discovery path packets to only those who are reliable, hence reducing the flooding. They also suggest that not keep the reliability or threshold value globally constant. As this value depends upon the nature of the communication, mobility of nodes and the nature of the network. Simulation shows that the results differ in lower and higher traffic. In the low traffic AODV and AODV-REL (their proposed protocol) works similar but as the traffic increases AODV-REL performs better than the simple AODV.

2.8. Problem statement

Path stability is the key issue common in all QoS models. No QoS model can be fully functionalized without achieving path stability. Therefore, path stability is key to all QoS models. If frequently route breakage occurs, frequent route discovery mechanism has to initiate and route discovery process is very expensive as it floods the network with control packets consuming lot of network resources.

There are different proposed protocols focuses on the route stability some are reactive in nature and some behave pro-actively. The reactive protocols e.g. AODV are fast enough in route discovery mechanism but lacks in providing stable routes. Therefore, route breakage is very frequently occurs in such protocols. To overcome this problem some enhancement were proposed in already existing protocols such as the mechanism of fast route repair e.g. Local repair procedure. However, local repair has also some limitations associated with it, they demand high node density and low traffic load. There even come such situations where local repair starts harming the network and cannot exploit the best available path that can be discovered only by the source node.

The other class of protocols providing path stability is of pro-active in nature. These protocols provides the stable path but consuming a lot of network resources in the mean time. These protocols maintain the network table on each node with the different information depending upon the technique implemented. Some stores the signal strength information of nodes while other stores the reliability factor of links etc. The main limitations of such protocols are the continuous consumption of network resources e.g. bandwidth, battery power etc for the updating of their network tables. The case becomes even worst when every node is maintaining the network information of all other nodes. It has been observed through different simulations results that a fair number of the nodes information kept in a node's network table remain unutilized throughout the whole network life. However, there is no other choice than to maintain the information of all nodes because one cannot predict that which node might be used in future.

Therefore, to address this problem some try to exploit the features of both reactive and pro-active approaches e.g. Stability aware cluster routing protocols [11]. In such kind of protocols, a single node that is cluster head has to maintain the whole information of the nodes present only in the given cluster and communication between clusters occurs reactively. The basic problem to this approach is very clear that a lot of load to maintain information is assigned to a single node and this cluster head is also responsible for the discovery of routes. Therefore, the battery power of such node will diminish very soon.

By keeping in view, the limitations associated with all the proposed protocols of either nature, our idea was to come up with such a hybrid protocol i.e. reactive in nature but can exploit the key benefits of pro-active protocols without consuming the network resources continuously

CHAPTER#3

METHODOLOGY

3. Methodology

3.1. History Based Stable Route Protocol (HBSR)

As dynamic topology is the main characteristic of MANETs, the main problem in MANETS is route breakage. Route breakage results in broadcast of route request packet in the network. This broadcast of request packet results in the network congestion, bandwidth utilization and propagation delay.

Proposed solution is based on breakege history of nodes. This movement history shows how many times a node has broken the route. Each node will maintain its own history rather than maintaining the history of all the network nodes on event driven basis. The history will be incremented by one whenever node breaks the route and decremented after certain time-period. A timer is attached with history entry

Whenever a node detects that another node has broken the route, the detecting node will broadcast this fact in the network. When this fact reaches the node who has broken the route will increment its history by one and reinitiate the timer while all other nodes will further broadcast the fact. The history will be decremented by one when timer will reach zero. When history of a node equals or exceeds threshold value it is declared notorious and will not be used in the route selection, as it will simply discard the request packet.

All the nodes in the network are classified into four classes of fairness based on their movement histories. Classes are as follows class A, class B, class C and class D. A node present in class A has less movement history and is more reliable than the node present in class B, similarly class B node is more reliable than the class C, so class A node is most reliable and the class C node is least reliable. The class D is composed of all notorious nodes.

Whenever a node wants a route to another node, it broadcasts a route request packet. This packet will probably reach each node in the network. A node which receives the request packet can belong to one of four categories. It may either be an intermediate node with no route to destination or it may be an intermediate node with a route to destination or it may be destination or even it may be notorious node.

The request packet consists of tuple-7 <req_id, source_IP, destination_IP, class_A, class_B, class_C, hop_count>. Req_id is used to distinguish a request from other request as at same time many request packets may be floating in the network. The reply packet consists of 7 tuples <req_id, reply_id source_IP, destination_IP, class_A, class_B, class_C, hop_count>.

HBSR can be defined as

```
HBSR (IN as Intermediate node, NN as Notorious node, RPR as Request packet received)

    If node_received = NOT (NN)   then

        If node_received = IN AND RPR = false

            then

                If node_received has route then

send_reply;

increment_history;

forward_request;

Else if node_recveived has no route   then
```

```
increment_history;
forward_request;
        Else If node_recveived is destination then
calculate_class_counter;
find_best_path;
send_reply;
  Else
discard_req;
  End if
```

In case the receiving node belongs to class D which is composed of notorious nodes, the request packet is discarded so that a notorious node will not be part of route.

In case, if the node receiving request packet is intermediate node and it has not received the same request packet earlier then it will increment the counter of either class to whom it belongs. After incrementing the field, it will further broadcast the packet. If its history is equal to or more than threshold history it will not further broadcast the packet. If this node again receives the request packet with same req_id, it will simply discard it to avoid network congestion. This discarding of request packet with same id will help in selecting multiple disjoint routes. Now if intermediate node has a route in its routing table it will send reply to source with available route. The reply packet also contains three fields for classes. When a route reaches to source from intermediate node or from destination, proposed protocol will start to transmit data which comes first and when other route reply comes either from destination or from intermediate node proposed protocol will compare these routes by their ratios and switch to best one.

If node receiving request packet is destination, then all request packets reaching to it are making disjoint route. All required information needed to select best route is available at destination node. Here HBSR protocol will select best route based on the movement history. As each request packet shows how many nodes belong to each class along with total hop count. The destination will compare the all class nodes with total hop count of the route of each request packet. First the counter of class C of each request packet is compared for all available routes. A route with less class C counter is more stable than a route with more class C counter. So a route with fewer counters will be selected. Here if two or more routes have same class C counter, then class B nodes counter for all packets will be compared. A route with less class B counter will be selected. If it is also same, then class A counters will be compared for decision-making. Now destination will send two disjoint best routes to the source. The reply packet has three fields for classes to show no of nodes in each class

CHAPTER#4
RESULTS

4. Results

We have simulated our proposed protocol in Ns2.28 using advanced server 4 update 3 (Red Hat Linux). The standard mobility model of 50 nodes is used. In this model grid (area for simulation) is 670*670 meter and 20 nodes move to other locations at the speed of 10m/sec. For transfer of data standard transmission model is used where maximum nodes are 50, 10 connections at a time are possible and nodes transfer data at 4Mbps. This model makes use of CBR (constant bit rate) for transmitting data. But on top of transmission layer we can use any application protocol as we also used, in addition, the FTP protocol. Other options for simulation are as

Channel type	Channel/WirelessChannel
Radio-propagation model	Propagation/TwoRayGround
Network interface type	Phy/WirelessPhy
MAC type	Mac/802_11
Interface queue type	Queue/DropTail/PriQueue
Ink layer type	LL
Antenna model	Antenna/OmniAntenna
Max packets in ifq	100
Simulation time	$5x10^3$ m/s

With these options and models when we start simulation, all the nodes are at their initial position set by mobility model as shown in the figure.

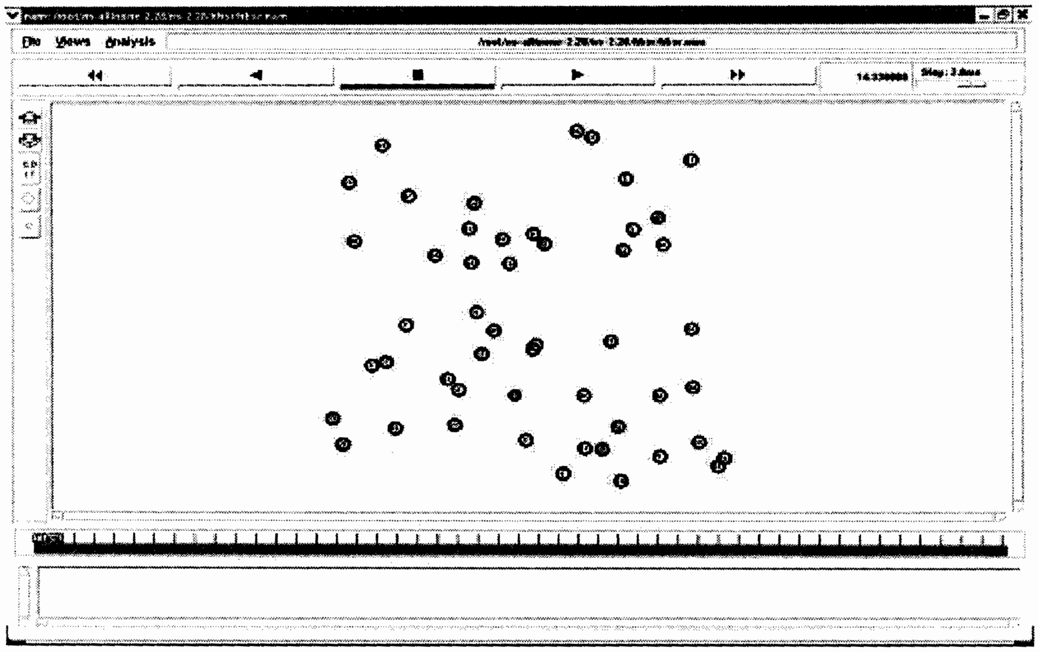

Figure 4.1: Initial Position of Nodes

In addition to data transmission file used we also have set a *ftp* connection between two nodes.

First of all nodes exchange history information with their neighbors is as shown in figure.

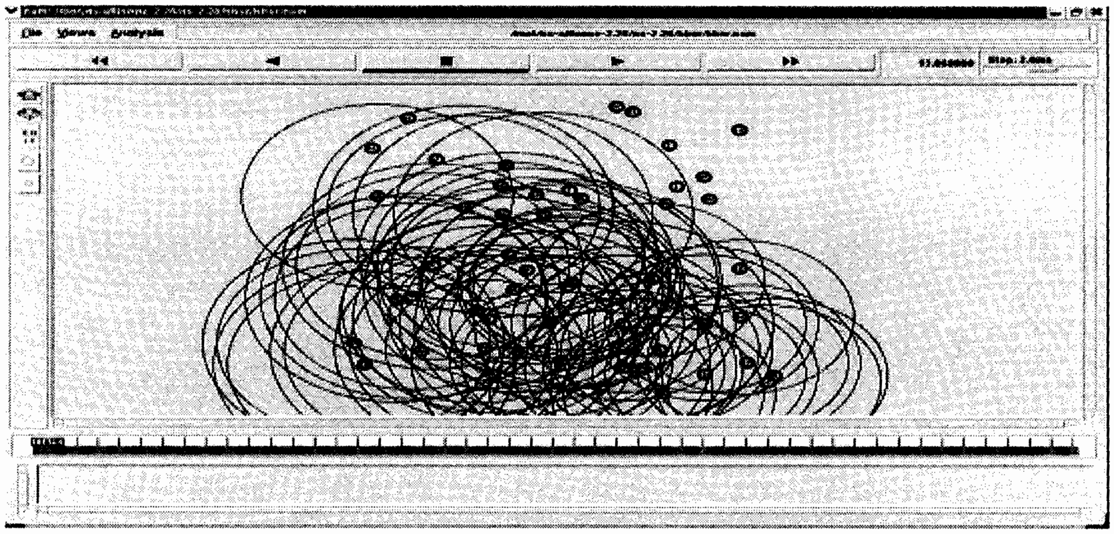

Figure 4.2: Nodes Exchange History with their Neighbors

Node 0 needs to send data to node 1 but node 0 and node 1 are not in transmission range of each other. So data transmitted by node 0 must be relayed on some nodes or node in order to reach node 1. Node 0 starts sending data to node 1 (ftp data) in form of packets as shown in figure

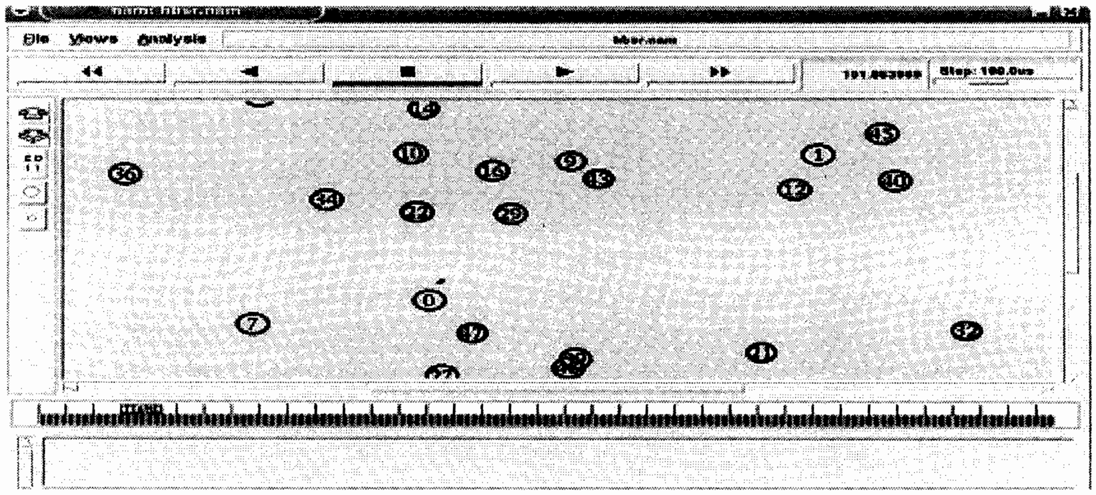

Figure 4.3: Transfer of Data between Nodes

Which are relayed at node 29 as shown in figure and figure

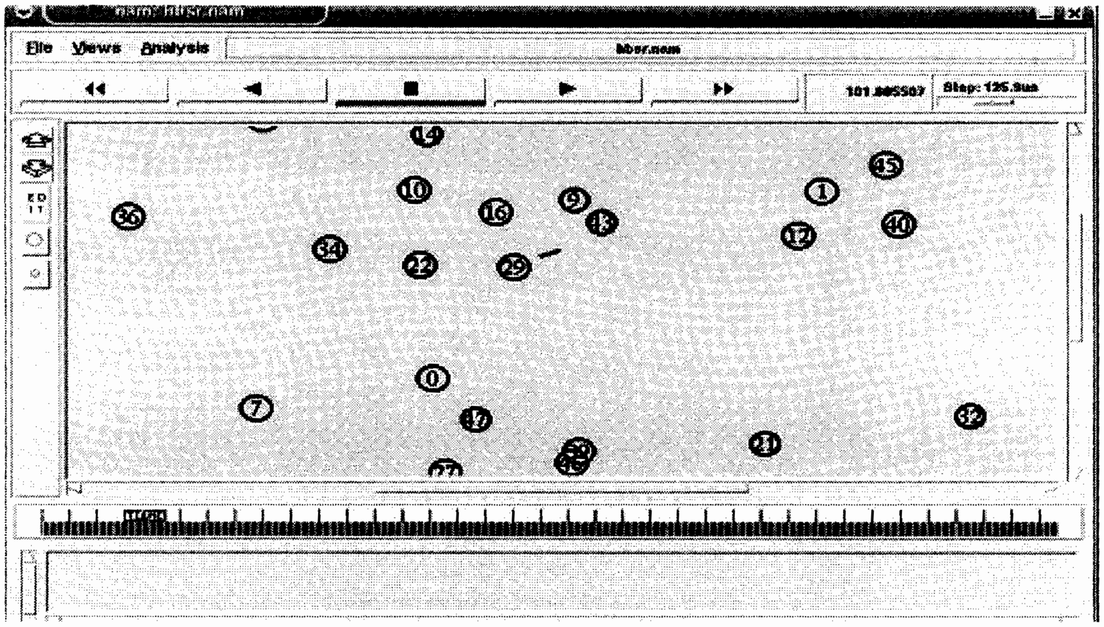

Figure 4.4: Node Relaying Data

Node 29 accepts data from node 0 relay this data then retransmit data to node 1. During this ftp transmission of data, CBR continues to transmit data among nodes. At time 600 m/s, nodes have moved from their initial position to positions as shown in figure

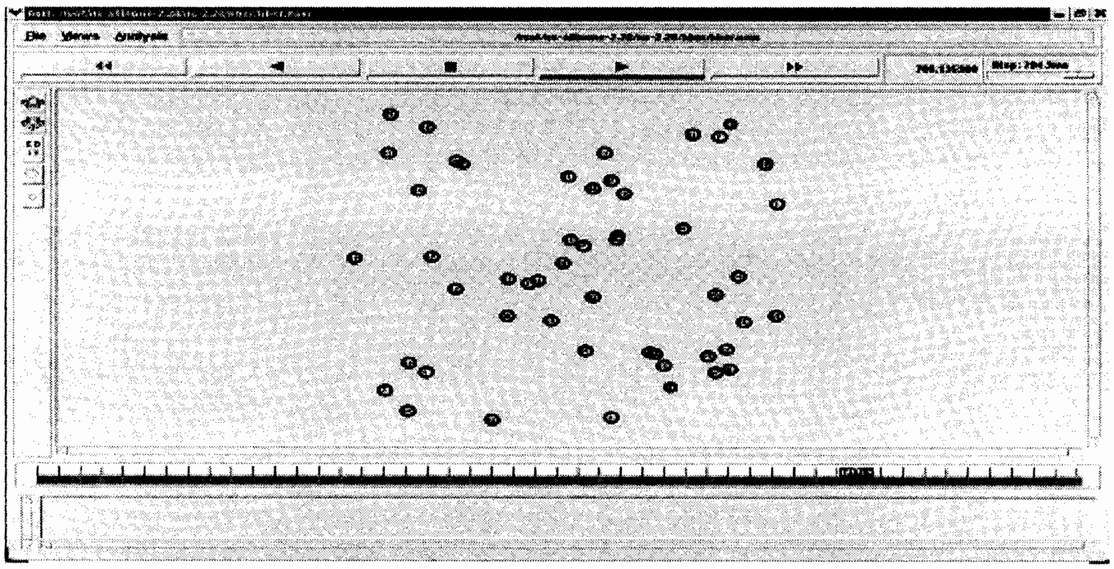

Figure 4.5: Movement of Nodes at 600m/s of Simulation Time

Even nodes have moved to other positions node 0 still transmit data as shown in figure

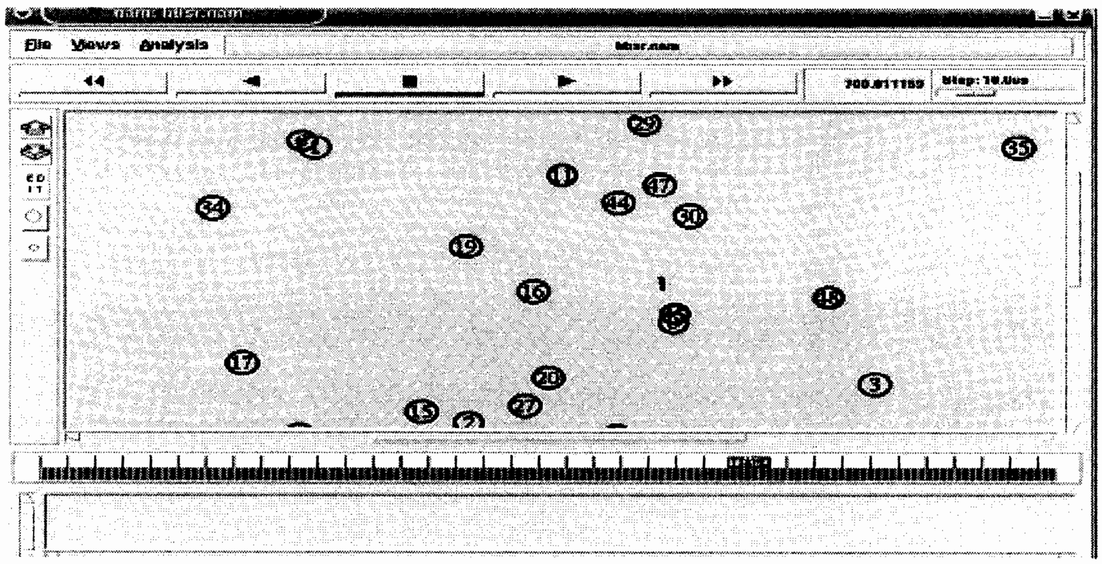

Figure 4.6: Transfer of Data after Node Mobility

We can show the functionality of HBSR by using trace file made by ns. Actually nam file is made from trace file, so results can be explained by trace file. If in the trace file formed by our simulation we track down to the time 100.000000, we see node 0 has send packet destined to node 1. This packet is received by node 21 at time 100.005092373 and at time 100.005092373, packet is forwarded by node 21. This forwarded packet is actually received by node 1 at time r 100.010328369. Node 1 then sends ACK of this packet at time 100.010328369. A preview of trace file is shown below.

```
s 100.000000000 _0_ AGT  --- 1031 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0]
[0 0] 0 2
r 100.000000000 _0_ RTR  --- 1031 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0]
[0 0] 0 2
s 100.000000000 _0_ RTR  --- 1031 tcp 60 [0 0 0 0] ------- [0:0 1:0 32 21]
[0 0] 0 2
r 100.005092373 _21_ RTR  --- 1031 tcp 60 [13a 15 0 800] ------- [0:0 1:0
32 21] [0 0] 1 2
f 100.005092373 _21_ RTR  --- 1031 tcp 60 [13a 15 0 800] ------- [0:0 1:0
31 1] [0 0] 1 2
r 100.010328369 _1_ AGT  --- 1031 tcp 60 [13a 1 15 800] ------- [0:0 1:0
31 1] [0 0] 2 2
s 100.010328369 _1_ AGT  --- 1032 ack 40 [0 0 0 0] ------- [1:0 0:0 32 0]
[0 0] 0 2
r 100.010328369 _1_ RTR  --- 1032 ack 40 [0 0 0 0] ------- [1:0 0:0 32 0]
[0 0] 0 2
s 100.010328369 _1_ RTR  --- 1032 ack 60 [0 0 0 0] ------- [1:0 0:0 32 43]
[0 0] 0 2
r 100.015702888 _43_ RTR  --- 1032 ack 60 [13a 2b 1 800] ------- [1:0 0:0
32 43] [0 0] 1 2
```

```
f 100.015702888 _43_ RTR  --- 1032 ack 60 [13a 2b 1 800] ------- [1:0 0:0
31 0] [0 0] 1 2
r 100.018200614 _0_ AGT  --- 1032 ack 60 [13a 0 2b 800] ------- [1:0 0:0
31 0] [0 0] 2 2
s 100.018200614 _0_ AGT  --- 1033 tcp 1040 [0 0 0 0] ------- [0:0 1:0 32
0] [1 0] 0 2
r 100.018200614 _0_ RTR  --- 1033 tcp 1040 [0 0 0 0] ------- [0:0 1:0 32
0] [1 0] 0 2
s 100.018200614 _0_ RTR  --- 1033 tcp 1060 [0 0 0 0] ------- [0:0 1:0 32
21] [1 0] 0 2
s 100.018200614 _0_ AGT  --- 1034 tcp 1040 [0 0 0 0] ------- [0:0 1:0 32
0] [2 0] 0 2
r 100.018200614 _0_ RTR  --- 1034 tcp 1040 [0 0 0 0] ------- [0:0 1:0 32
0] [2 0] 0 2
s 100.018200614 _0_ RTR  --- 1034 tcp 1060 [0 0 0 0] ------- [0:0 1:0 32
21] [2 0] 0 2
r 100.028458916 _21_ RTR  --- 1033 tcp 1060 [13a 15 0 800] ------- [0:0
1:0 32 21] [1 0] 1 2
f 100.028458916 _21_ RTR  --- 1033 tcp 1060 [13a 15 0 800] ------- [0:0
1:0 31 1] [1 0] 1 2
r 100.038437058 _1_ AGT  --- 1033 tcp 1060 [13a 1 15 800] ------- [0:0 1:0
31 1] [1 0] 2 2
s 100.038437058 _1_ AGT  --- 1035 ack 40 [0 0 0 0] ------- [1:0 0:0 32 0]
[1 0] 0 2
r 100.038437058 _1_ RTR  --- 1035 ack 40 [0 0 0 0] ------- [1:0 0:0 32 0]
[1 0] 0 2
s 100.038437058 _1_ RTR  --- 1035 ack 60 [0 0 0 0] ------- [1:0 0:0 32 43]
[1 0] 0 2
r 100.040874566 _43_ RTR  --- 1035 ack 60 [13a 2b 1 800] ------- [1:0 0:0
32 43] [1 0] 1 2
```

To explain trace file we pick the time slot given below

```
r 100.005092373 _21_ RTR  --- 1031 tcp 60 [13a 15 0 800] ------- [0:0 1:0
32 21] [0 0] 1 2
```

First three fields show that node **21** is receiving packet at time **100.005092373, RTR** is signal that node is 'Ready To Receive'. **1031** is the unique id of packet received whereas **tcp** represents type of packet. Common header size of packet is **60**. **13a** is the expected time to send packet on wireless channel Mac-id of receiving node is **15** while that of sending node is **0. 800** is a fixed value which represent ETHERTYPE_ARP. Next **bracket [0:0 1:0 32 21]** shows 3 tier IP information of both source and destination. **[0 0]** shows that seqno and ACK are **0**. **1** is the destination while **2** shows no of hops to reach destination.

CHAPTER#5

CONCLUSION & FUTURE WORK

5. Conclusion and Future Work

5.1. Conclusion

Mobile Ad-hoc Networks is the emerging field in computer networks. MANETs are difficult to handle because of their some basic features and they are widely used due to their applications. Routing issues, security issues and QoS issues in the MANETs are also discussed.

As the primary focus was on the path stability, that is the most important issue in any of the QoS model. Some common protocols designed for path stability are also discussed. These protocols are the extensions of reactive, pro-active and hybrid approaches. Some focus on the fast link repair procedure in case of link breakage and some on the selection of stable behavior nodes in route selection in order to minimize the chances of link breakage

In the last section, we present problem that for achieving any QoS model route stability is the main factor. If we succeeded in achieving route stability we can achieve QoS routing. Our proposed protocol named "history based stable route protocol (HBSR) "will provide route stability. By selecting stable routes, following benefits will be achieved.

i) Reduces consumption of bandwidth for control traffic
ii) Reduces delay during transmission
iii) Provides multiple disjoint path
iv) Provides benefits of both reactive and proactive protocols

5.2. Future Work

On destination, different selection criteria can be implemented on the basis of available information. A more sophisticated selection criteria can be worked out which also weigh no of hops factor but main focus still remain on stability of routes not the simple hop count. But this can be more efficient than suggested one in HBSR in case where a route with greater hop count is selected even though a route with very less hop count is available. Secondly, a formula can be derived for threshold history based on the network parameters such that no of nodes in the network, movement pattern, total length (in kilometers) of network and signal strength. May Allah give us strength and courage to do that (Ameen)

Appendix A

APPENDIX A
Simulation Code

```
//include files for the node agent



#include <tcl.h>                // for tcl linkage
#include "agent.h"                    // inherit from base agent
#include <stdio.h>
#include <hbsr/hbsr.h>







// include files for the packet header
#include "hbsr.h"

/////////////////////////////////////////////////////////////////////

int hdr_hbsr::offset_;

static class HbsrHeaderClass : public PacketHeaderClass {
public:
    HbsrHeaderClass() : PacketHeaderClass("PacketHeader/Hbsr", sizeof(hdr_hbsr)) {
        bind_offset(&hdr_hbsr::offset_);
    }
} class_hbsr_hdr;

//////////////////////////////////////////////////////////////////////

static class HbsrAgentClass : public TclClass {
public:
    HbsrAgentClass() : TclClass("Agent/HBSR") {}
    TclObject* create(int argc, const char*const* argv) {
        return(new HbsrAgent((nsaddr_t) Address::instance().str2addr(argv[4])));
    }
} class_hbsr_agent;

/////////////////////////////////////////////////////////////////////

void helloTimer::expire(Event* event)
{
      agent_->firsttimehello=0;
      agent_->sendHello();
      resched(3);
}
/////////////////////////////////////////////////////////////////////

void routeTimer::expire(Event* event)
{
     int sourceid;
     int backpointer[2];
     int hopcount[2];
     int class_A[2];
     int class_B[2];
     int class_C[2];

     int index = agent_->findbestroute();
     sourceid=agent_->sourced[agent_->source_descrip_size].sourceid;
backpointer[0]=agent_->sourced[agent_->source_descrip_size].r_d[index].backpointer;
     hopcount[0]=agent_->sourced[agent_->source_descrip_size].r_d[index].hopcount;
     class_A[0]=agent_->sourced[agent_->source_descrip_size].r_d[index].class_A;
     agent_->sourced[agent_->source_descrip_size].r_d[index].class_A=10000;
     class_B[0]=agent_->sourced[agent_->source_descrip_size].r_d[index].class_B;
     agent_->sourced[agent_->source_descrip_size].r_d[index].class_B=10000;
     class_C[0]=agent_->sourced[agent_->source_descrip_size].r_d[index].class_C;
     agent_->sourced[agent_->source_descrip_size].r_d[index].class_C=10000;

     Packet * hbsr1 = agent_->createHbsrPkt();
     hdr_hbsr* hbs = HDR_HBSR(hbsr1);
     hbs->type = REPLY;
     hbs->source_id = agent_->id;
     hbs->dest_id = sourceid;
     hbs->current_id = backpointer[0];
     hbs->hopcount =hopcount[0];
     hbs->class_A =class_A[0];
     hbs->class_B =class_B[0];
     hbs->class_C =class_C[0];

     HDR_CMN(hbsr1)->size_ = IP_HDR_LEN;
     agent_->send(hbsr1, NULL);

     index = agent_->findbestroute();
     backpointer[1]=agent_->sourced[agent_-
>source_descrip_size].r_d[index].backpointer;
     hopcount[1]=agent_->sourced[agent_->source_descrip_size].r_d[index].hopcount;
     class_A[1]=agent_->sourced[agent_->source_descrip_size].r_d[index].class_A;
     class_B[1]=agent_->sourced[agent_->source_descrip_size].r_d[index].class_B;
     class_C[1]=agent_->sourced[agent_->source_descrip_size].r_d[index].class_C;

     hbs->source_id = agent_->id;
     hbs->dest_id = sourceid;
     hbs->current_id = backpointer[1];
     hbs->hopcount =hopcount[1];
     hbs->class_A =class_A[1];
     hbs->class_B =class_B[1];
     hbs->class_C =class_C[1];

     HDR_CMN(hbsr1)->size_ = IP_HDR_LEN;
     agent_->send(hbsr1, NULL);


     agent_->clearall();
}
```

```
//////////////////////////////////////////////////////////////////////
/// implementation of the hbsr agent class

// constructor
HbsrAgent::HbsrAgent(nsaddr_t idp) : Agent(PT_HBSR), helloTimer_(this),
routeTimer_(this)
{
      printf("shams here\n");
      id = idp;
      for (int i = -1 ; i < 50 ; i++)
            neighbour[i] = -1;
      neighbourcount=0;
      history = 0;
      firsttimehello=1;
        routesize=0;
      source_descrip_size=0;
      for (int i = 0 ; i < 10 ; i++)
            sourced[i].r_d_size=0;
}

// receive packet of the function
void HbsrAgent::recv(Packet* incomingPkt, Handler*)
{

    // process incomingPkt according to its type
    hdr_hbsr* hbsrHdr = HDR_HBSR(incomingPkt);

    switch (hbsrHdr->type)
    {

        case HELLO:
            handleHello(incomingPkt);

        case REQUEST:
            handleRequest(incomingPkt);
            break;
        case REPLY:
            handleReply(incomingPkt);
            break;
        case HISTORY:
            handleHistory(incomingPkt);
            break;
        default:
            Packet::free(incomingPkt);
            break;
    }
}

int HbsrAgent::command(int argc, const char*const* argv)
{

    Tcl& tcl = Tcl::instance();
    printf ("shams here\n");

    if (2 == argc)
      {
        if (0 == strcasecmp(argv[1], "init-node"))
        {
            initNode();
            return TCL_OK;
        }
          else if (0 == strcasecmp(argv[1], "start-node"))
        {
              startNode();
            return TCL_OK;
        }
        else if (0 == strcasecmp(argv[1], "stop-node"))
        {
              stopNode();
            return TCL_OK;
        }
        else if (0 == strcasecmp(argv[1], "hello"))
        {
              sendHello();
            return TCL_OK;
        }



  }

else if (3 == argc)
  {


            if (0==strcasecmp(argv[1], "route-request"))
            {
              int source = id;
              int dest = (int)argv[1];
              establishroutes (source,dest);
            }
            if (0==strcasecmp(argv[1], "on-node"))
            {
              node_ = (MobileNode *)tcl.lookup(argv[2]);
            if (NULL == node_)
            {
              tcl.resultf("cannot attach node to this               agent.\n");
              return (TCL_ERROR);
            }
                  return TCL_OK;
            }
        else if(strcmp(argv[1], "log-target") == 0 ||
              strcmp(argv[1], "tracetarget") == 0)
        {

            logtarget = (Trace*) TclObject::lookup(argv[2]);
            if(logtarget == 0)
              return TCL_ERROR;
            return TCL_OK;
        }
        else if (strcmp(argv[1], "port-dmux") == 0)
```

```
            {
                  dmux_ = (PortClassifier *)TclObject::lookup(argv[2]);

                  if (dmux_ == 0)
                {
                  fprintf (stderr, "%s: %s lookup of %s failed\n",
                  __FILE__,argv[1], argv[2]);
                  return TCL_ERROR;
                }
                return TCL_OK;
            }
            else if(strcmp(argv[1], "drop-target") == 0)
            {
            //    int stat = rqueue.command(argc,argv);
            //      if (stat != TCL_OK) return stat;

                  return Agent::command(argc, argv);
            }
        }


    return Agent::command(argc, argv);
}

void HbsrAgent::sendHello()
{
      Packet * hbsr1 = createHbsrPkt();
      hdr_hbsr* hbs = HDR_HBSR(hbsr1);
      hbs->type = HELLO;
      hbs->source_id = id;
      HDR_CMN(hbsr1)->size_ = IP_HDR_LEN;
      send(hbsr1, NULL);

}

void HbsrAgent::initNode()
{}

void HbsrAgent::startNode()
{
      printf("shams here\n");
}

void HbsrAgent::stopNode()
{}

char HbsrAgent::getclass (int history)
{
      char returnchar;
      switch (history)
      {
            case 0:case 1:
                  returnchar = 'A';
            break;
            case 2:case 3:
                  returnchar = 'B';
            break;
            case 4:case 5:
                  returnchar = 'C';
            break;
      }

      return returnchar;
}

void HbsrAgent::handledestination(Packet* incomingpkt)
{
      int s_id = HDR_HBSR(incomingpkt)->source_id;
      int present = 0 ;
      int index;
      for (int i = 0 ; i < source_descrip_size ; i++)
            {
                  if (s_id == sourced[i].sourceid)
                  {
                        present=1;
                        index = i;
                  }
            }
      if (present) // add route
      {
            sourced[index].r_d[sourced[index].r_d_size].backpointer =
HDR_HBSR(incomingpkt)->current_id;
            sourced[index].r_d[sourced[index].r_d_size].hopcount =
(HDR_HBSR(incomingpkt)->hopcount++);
            char cl = getclass (history);
            sourced[index].r_d[sourced[index].r_d_size].class_A = HDR_HBSR(incomingpkt)-
>class_A;
            if (cl == 'A')
                  sourced[index].r_d[sourced[index].r_d_size].class_A++;
            sourced[index].r_d[sourced[index].r_d_size].class_B = HDR_HBSR(incomingpkt)-
>class_B;
            if (cl == 'B')
                  sourced[index].r_d[sourced[index].r_d_size].class_B++;
            sourced[index].r_d[sourced[index].r_d_size].class_C = HDR_HBSR(incomingpkt)-
>class_C;
            if (cl == 'C')
                  sourced[index].r_d[sourced[index].r_d_size].class_C++;
            sourced[index].r_d_size++;
      }
      else // add source
      {
            char cl = getclass (history);
            sourced[source_descrip_size].sourceid = HDR_HBSR(incomingpkt)->source_id;
      sourced[source_descrip_size].r_d[sourced[source_descrip_size].r_d_size].backpointe
r = HDR_HBSR(incomingpkt)->current_id;
sourced[source_descrip_size].r_d[sourced[source_descrip_size].r_d_size].hopcount =
(HDR_HBSR(incomingpkt)->hopcount++);
sourced[source_descrip_size].r_d[sourced[source_descrip_size].r_d_size].class_A =
HDR_HBSR(incomingpkt)->class_A;
sourced[source_descrip_size].r_d[sourced[source_descrip_size].r_d_size].class_B =
HDR_HBSR(incomingpkt)->class_B;
```

```
sourced[source_descrip_size].r_d[sourced[source_descrip_size].r_d_size].class_C =
HDR_HBSR(incomingpkt)->class_C;
      if (cl == 'A')

      sourced[source_descrip_size].r_d[sourced[source_descrip_size].r_d_size].class_A++;

      if (cl == 'B')

      sourced[source_descrip_size].r_d[sourced[source_descrip_size].r_d_size].class_B++;

      if (cl == 'C')

      sourced[source_descrip_size].r_d[sourced[source_descrip_size].r_d_size].class_C++;

      sourced[source_descrip_size].r_d_size++;
      source_descrip_size++;


      }

}

int HbsrAgent::routepresent(Packet* incomingPkt)
{
      for (int i = 0 ; i < routesize ; i++)
      {
            if (HDR_HBSR(incomingPkt)->source_id == routes_[i].source &&
HDR_HBSR(incomingPkt)->dest_id == routes_[i].dest)
                  return 1;
      }
      return 0;
}
void HbsrAgent::handleRequest (Packet* incomingPkt)
{
      if (history < THRESHOLD)
      {
            if (HDR_HBSR(incomingPkt)->dest_id == id)
            {
                  handledestination(incomingPkt);
                  Packet::free(incomingPkt);
            }
            else if (!routepresent(incomingPkt))
            {
                  routes_[routesize].source = HDR_HBSR(incomingPkt)->source_id;
                  routes_[routesize].dest = HDR_HBSR(incomingPkt)->dest_id;
                   routes_[routesize].backpointernode = HDR_HBSR(incomingPkt)-
>current_id;
                  routesize++;
                  char myclass = getclass(history);
                  hdr_hbsr* hbs = HDR_HBSR(incomingPkt);
                  hbs->current_id = id;
                  hbs->hopcount++;
                  if (myclass == 'A')
                        hbs->class_A++;
                  else if (myclass == 'B')
                        hbs->class_B++;
```

```
                else if (myclass == 'C')
                      hbs->class_C++;

                HDR_CMN(incomingPkt)->size_ = IP_HDR_LEN;
                send(incomingPkt, NULL);

            }
            else
            {
                Packet::free(incomingPkt);
            }

        }
        else
        {
            Packet::free(incomingPkt);
        }

}
void HbsrAgent::handleReply (Packet* incomingpkt)
{}
void HbsrAgent::handleHistory(Packet* incomingPkt)
{
        int presentflag=0;
        int saveindex=0;
        if (HDR_HBSR(incomingPkt)->source_id == id)
              history++;
        for (int i = 0 ; i < neighbourcount ; i++)
        {
              if (neighbour[i] == HDR_HBSR(incomingPkt)->source_id)
              {
                    presentflag=1;
                    saveindex=i;
              }
        }
        if (presentflag==1)
        {
              int temparray[50];
              for (int j = 0 ; j < neighbourcount ; j++)
              {
                    if (j != saveindex)
                          temparray[j] = neighbour[j];
              }
              for (int j = 0 ; j < (neighbourcount-1) ; j++)
                    neighbour[j] = temparray[j];
              neighbourcount--;
        }

          Packet::free(incomingPkt);

}

void HbsrAgent::handleHello(Packet* incomingPkt)
{
        if (firsttimehello=1)
        {
            if (neighbourcount == 0)
            {
                  helloTimer_.sched (3);  // set timer to expire
      //after 3 sec
            }
      neighbour[neighbourcount] = HDR_HBSR(incomingPkt)->source_id;
      neighbourcount++;
      }
      else
      {
            int flag=0;
            for (int j = 0 ; j < neighbourcount ; j++)
            {
                  if (neighbour[j] ==HDR_HBSR(incomingPkt)->source_id)
                  {
                        flag=1;
                  }
            }
            if (flag==0)
            {
                  neighbourcount++;
                 neighbour[neighbourcount]=HDR_HBSR(incomingPkt)->source_id;
            sendHistorypacket(HDR_HBSR(incomingPkt)->source_id);
            sendHistorypacket(HDR_HBSR(incomingPkt)->source_id);
            }
      }
        Packet::free(incomingPkt);
}

void HbsrAgent::sendHistorypacket(int sourceid)
{

      Packet * hbsr1 = createHbsrPkt();
      hdr_hbsr* hbs = HDR_HBSR(hbsr1);
      hbs->type = HISTORY;
      hbs->source_id = sourceid;
      HDR_CMN(hbsr1)->size_ = IP_HDR_LEN;
      send(hbsr1, NULL);
}

int HbsrAgent::findbestroute()
{

}

int HbsrAgent::clearall()
{

}
void HbsrAgent::establishroutes (int source, int dest)
{
      Packet * hbsr1 = createHbsrPkt();
      hdr_hbsr* hbs = HDR_HBSR(hbsr1);
      hbs->type = REQUEST;
      hbs->source_id = id;
      hbs->current_id = id;
```

```
        hbs->dest_id = dest;
        hbs->hopcount = 0;
        hbs->class_A=0;
        hbs->class_B=0;
        hbs->class_C=0;
        char cl = getclass (history);
        switch (cl)
        {
              case 'A':
                            hbs->class_A++;
              break;
              case 'B':
                            hbs->class_B++;
              break;
              case 'C':
                            hbs->class_C++;
              break;

        }

        HDR_CMN(hbsrl)->size_ = IP_HDR_LEN;
        send(hbsrl, NULL);
  }
```

```
# ======================================================================
# Define options
# ======================================================================
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)          Phy/WirelessPhy            ;# network interface type
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
#set opt(cp)            "../mobility/scene/cbr-50-20-4-512"
#set opt(sc)            "../mobility/scene/scen-670x670-50-600-20-0"
set val(ifqlen)         50                         ;# max packet in ifq
set val(nn)             50                         ;# number of mobilenodes
set val(rp)             HBSR                       ;# routing protocol
set val(topox)          670                     ;# topology width
set val(topoy)          670                     ;# topology height
#set opt(stop)          1000.000000000000
# ======================================================================
# Main Program
# ======================================================================


#
# Initialize Global Variables
#
set ns_         [new Simulator]
set tracefd     [open simple.tr w]
$ns_ trace-all $tracefd

set namtrace    [open hbsr.nam w]
$ns_ namtrace-all-wireless $namtrace $val(topox) $val(topoy)


# set up topography object
set topo       [new Topography]

$topo load_flatgrid 670 670

#
# Create God
#
create-god $val(nn)
set god_ [God instance]

#
#  Create the specified number of mobilenodes [$val(nn)] and "attach" them
#  to the channel.
#  Here two nodes are created : node(0) and node(1)

# configure node

        $ns_ node-config -adhocRouting HBSR \
                         -llType $val(ll) \
                         -macType $val(mac) \
                         -ifqType $val(ifq) \
                         -ifqLen $val(ifqlen) \
                         -antType $val(ant) \
                         -propType $val(prop) \
                         -phyType $val(netif) \
                         -channelType $val(chan) \
                         -topoInstance $topo \
                                  -agentTrace ON \
                                  -routerTrace ON \
                                  -macTrace OFF \
                                  -movementTrace OFF

        for {set i 0} {$i < $val(nn) } {incr i} {
                set node_($i) [$ns_ node]
                #$node_($i) random-motion 0             ;# disable random motion
                #set hbsragent_($i) [new Agent/DSDV]
                #$node_($i) attach $hbsragent_($i) 217
        #       $hbsragent_($i) start-node
       #        $hbsragent_($i) on-node $node_($i)

        }

# Source the Connection and Movement scripts

if { $opt(cp) == "" } {
        puts "*** NOTE: no connection pattern specified."
        set opt(cp) "none"
} else {
        puts "Loading connection pattern..."
        source $opt(cp)
}

if { $opt(sp) == "" } {
        puts "*** NOTE: no movement pattern specified."
        set opt(cp) "none"
} else {
        puts "Loading movement pattern..."
        source $opt(sc)
}


#
# Tell nodes when the simulation ends
#
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at 1000.000000000000 "$node_($i) reset";
}
$ns_ at 1000.000000000000 "stop"
#$ns_ at  $opt(stop)    "$ns_ nam-end-wireless $opt(stop)"
$ns_ at 1000.000000000001 "puts \"NS EXITING...\" ; $ns_ halt"

proc stop {} {
    global ns_ tracefd
    $ns_ flush-trace
    close $tracefd

    global ns_ namtrace;
    $ns_ flush-trace
    close $namtrace;
}

puts "Starting Simulation..."
$ns_ run
```

Appendix B

APPENDIX B

References

[1] Alejandro Quintero, Samuel Pierre, Benjamin Macabeo, A routing protocol based on node density for ad hoc networks, Ad Hoc Networks 2 (2004) 335–349.

[2] Charles Perkins, Ad-Hoc On Demand Distance Vector (AODV) Routing: draft-ietf-manet-aodv-00.txt, 20 November 1997.

[3] Youngki Hwang and Pramod Varshney, Fellow, IEEE, An Adaptive QoS Routing Protocol with Dispersity for Ad-hoc Networks, Proceedings of the 36th Hawaii International Conference on System Sciences: (HICSS'03) 0-7695-1874-5/03 $17.00 © 2002 IEEE

[4] Chun-Yuah Chiu, Eric Hsiao-kuang Wu, and Gen-Huey Chen, Stability Aware Cluster Routing Protocol for mobile Ad-Hoc Networks., Proceedings of the Ninth International Conference on Parallel and Distributed Systems (ICPADS'02) 1521-9097/02 © 2002 IEEE.

[5] Michael Gerharz, Christian de Waal, Peter Martini and Paul James, Strategies for Finding Stable Paths in Mobile Wireless Ad Hoc Networks, Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks (LCN'03) 0742-1303/03 © 2003 IEEE.

[6] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario-based performance analysis of routing protocols for mobile ad-hoc networks. In 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom), pages 195.206, Seattle Washington, August 1999.

[7] Robert Radziwilowicz, a presentation "Quality of Service in Ad Hoc Networking" SITE, University of Ottawa, 05 November 2004.

[8] Harpreet S. Arora, *Towards Achieving QoS Guarantees in Mobile Ad- Hoc Networks*, M.S. thesis, Drexel University, October 2003

[9] Aura Ganz, *Quality of Service Provision in Mobile Ad hoc Networks* (MANET), University of Massachusetts, January 14, 2002.

[10] P.Jacquet, P.muhlethar, T.clausen, Laouiti,A.Qayyum, L.viennot Optimized link state routing protocol for adhoc network

[11] P.Jacquet, T.clausen rfc3626 Project Hipercom, INRIA ,October 2003

12] Manel Guerrero Zapata and N. Asokan, *Securing Ad hoc Routing Protocols*, 2002 ACM 1-58113-585-8/02/0009.

[13] Hongmei Deng, Wei Li, and Dharma P. Agrawal, *Routing Security in*

Wireless Ad Hoc Networks, IEEE Communications Magazine • October 2002, 0163-6804/02/© 2002 IEEE

[14] Chenxi Zhu and M. Scott Corson, *QoS routing for mobile ad hoc networks*, 0-7803-7476-2/02/(c) 2002 IEEE

[15] Vartika Bhandari, Nupur Kothari and Dheeraj Sanghi, *A Reliable On-demand Routing Paradigm for Mobile Ad hoc Networks*, Department of Computer Science & Engineering, Indian Institute of Technology Kanpur, INDIA.

[16] Ian D. Chakeres, Elizabeth M. Belding-Royer, *"AODV Routing Protocol Implementation Design"*, University of California, Santa Barbara

PUBLICATION

History Based Stable Routes Protocol

Muhammad Shams-Ul-Haq, Kashif Sajjad Bhatti, Sikander Hayat Khiyal
Department of Computer Science, Faculty of Applied Sciences,
International Islamic University Islamabad.
Shams_5@yahoo.com, kashifsajad@gmail.com, hdcs@iiu.edu.pk

Abstract

MANETs is a network without any fixed infrastructure. Each node act as host as well as router to relay the packets destined for other nodes. Due to this characteristic of this network, the route breakage is very frequent. To solve this problem, we select the route which shows stable behavior. Our proposed solution is based on the movement history of the node. By selecting those nodes which have fair movement history, route breakage will be reduced significantly. Theoretical computation shows that routes selected on the basis of movement history are long lived.

1. Introduction

Mobile Adhoc Networks (MANETS) have been the focus of many recent research and development efforts. A Mobile Adhoc Network (MANET) is mobile wireless network composed of several mobile nodes, likely to communicate among themselves without the intervention of any centralized management or existing infrastructure. Projected Examples of such networks includes: defense based application like War scenarios, Disaster Relief Operations like Earthquake, Rural Areas and Commercial Applications like Home networking extending internet connectivity. One of the reasons for the considerable effort in this area is that routing in such networks is a very challenging task, since network topology changes occur frequently (Routing Issues). Due to these frequent changes an established route between nodes may break. If this happens then data packets must be Re-routed quickly to make path stable. There are many other issues regarding Mobile Adhoc Networks, i.e. Security, Quality of Service, battery power constraints. But in the design of most Adhoc Network Protocols, issue of path stability (Routing) has not been addressed. Only a few protocols aim at establishing stable routes but they depend on specific hardware features or an unverified assumption. [1,5, 6, 7,8]

This paper presents a method which improves the path stability in MANETS. In section 2, we present previous work in the area of path stability. In section 3, we describe our idea of improving path stability in MANETS; results are in section 4 while conclusion is section 5.

2. Related Work

Most of the MANETs protocols can be classified as *best effort* protocols. Best effort protocols in MANETs can be divided into *reactive* and *proactive* protocols. *Reactive protocols* work on demand basis; when a route to a destination is required then it is established. These protocols reduce the bandwidth consumption as there is no need to store information in the tables for the nodes. Whereas *proactive protocols* are table-driven protocols where few tables are used to store the information of nodes. Proactive protocols maintain a route from each node to all other nodes in the network whereas reactive protocols do not. So reactive protocols impose initial delay but proactive protocols have no initial delay [1]. Examples of reactive protocols are ABR, AODV, CEDAR, DREAM, DSR, FORP, GEDIR, LAR, SSR, WAR whereas DSDV, GSR, WRP, FSR, GPSR and LANMAR are examples of proactive class [6].

Many strategies are introduced by the researchers to stabilize the route between source and destination which not only take into account the hop count but some other parameters also to select stable routes. AODV is the main on demand distance vector routing protocol. AODV discover route when it is required [2]. [1] Introduces the new concepts of local repair for AODV. In case of route breakage during transmission, broken route will be repaired. This repair procedure can be performed either by the original source node or by the intermediate node who has detected the route breakage. This route repair process consists of rediscovery of route to the destination. If this rediscovery is initiated by the original source of transmission then it is global repair. In this case the node who has detected the route breakage will send MAC layer intimation to the source that route has been broken. In the response source will start the discovery process for new route. Whereas in local repair the node that discovered the route breakage starts the rediscovery process of route to the destination. To increase the chances of local repair, a node with highest no of neighbors is selected in route discovery phase so that this node can perform local repair. This technique of local route repair can reduces the bandwidth and delay in transmission. But in some cases it can harm network by selecting a route which is very long even though original source has a better choice of route.

[4] Divides the network into several non-overlapping clusters and adopts proactive routing protocol inside the clusters and reactive routing protocol outside the clusters. The communication path between two nodes may be through many clusters. This distributes the maintenance efforts into many clusters. This protocol is divided into three parts; cluster head selection, cluster construction and cluster maintenance. Nodes with "max local degree" become cluster head forming stable clusters. In this mechanism each node which is not part of any cluster send control packet to all its neighbors telling them its node density. In response, all nodes that are not part of any cluster will send their density and a NODE with highest degree becomes the cluster head. Cluster head broadcast another control packet to form gravitational area. All nodes in the same gravitational area form the cluster with unique routing id. Broadcast by a node is restricted to its gravitational area i.e. its cluster. If a node want to communicate with a node that is registered in another cluster, this node has to contact its own head as outside the cluster only head can communicate as only heads maintain topological tables. If a node moves during transmission, this node will be put outside the cluster where it waits for a time to join any cluster, and in case, link breakage occurs between the inter-cluster nodes, then the cluster head will select another node to re-construct the path. This protocol works best in such networks where the node density is high and the mobility of nodes is low.

[3] Presents interesting technique in order to select the best stable path between source and destination. This technique is based on node's signal strength. The authors in this paper classify the nodes in first class, second class and third class nodes according to their signal strength. On the basis of these classes, links and routes are classified accordingly. This proposed protocol selects two disjoint paths for communication. Each node maintains network table, which contains updated lists of its neighbor, updated signal strength and minimum signal strength to transfer data. When the signal strength of a route becomes very week protocol switches the communication to second path before first route becomes unavailable. This preemptive switching reduces communication delay occurred due to route breakage during communication. But on the other hand it is also expensive in terms of bandwidth as maintaining of network tables consume continuously bandwidth.

3. History Based Stable Routes Protocol

As dynamic topology is the main characteristic of MANETs, the main problem in MANETS is route breakage. Route breakage results in broadcast of route request packet in the network. This broadcast of request packet results in the network congestion, bandwidth utilization and propagation delay.

Proposed solution is based on movement history of nodes. This movement history shows how many times a node has broken the route. Each node will maintain its own history rather than maintaining the history of all the network nodes on event driven basis. The history will be incremented by one whenever node breaks the route and decremented after certain time-period. A timer is attached with history entry

Whenever a node detects that another node has broken the route, the detecting node will broadcast this fact in the network. When this fact reaches the node who has broken the route will increment its history by one and reinitiate the timer while all other nodes will further broadcast the fact. The history will be decremented by one when timer will reach zero. When history of a node equals or exceeds threshold value it is declared notorious and will not be used in the route selection, as it will simply discard the request packet.

All the nodes in the network are classified into four classes of fairness based on their movement histories. Classes are as follows class A, class B, class C and class D. A node present in class A is more reliable than the node present in class B, similarly class B node is more reliable than the class C, so class A node is most reliable and the class C node is least reliable. The class D is composed of all notorious nodes.

Whenever a node wants a route to another node, it broadcasts a route request packet. This packet will probably reach each node in the network. A node which receives the request packet can belong to one of four categories. It may either be an intermediate node with no route to destination or it may be a intermediate node with a route to destination or it may be destination or even it may be notorious node.

The request packet consists of 7 tuples *<req_id, source_IP, destination_IP, class_A, class_B, class_C, hop_count>*. *Req_id* is used to distinguish a request from other request as at same time many request packets may be floating in the network. *The* reply packet consists of 7 tuples *<req_id, source_IP, destination_IP, class_A_ratio, class_B_ratio, class_C_ratio, hop_count>*.

```
HBSR can be defined as
HBSR (IN as Intermediate node, NN as Notorious
node, RPR as Request packet received)
    If node_received = NOT (NN)  then
        If node_received = IN AND RPR = false
        then
                If node_received has route then
                        send_reply;
                    increment_history;
                     forward_request;
                Else if node_recveived has no route
                then
                    increment_history;
                     forward_request;
        Else If node_recveived is destination then
                       calculate_ratio;
                       find_best_path;
                         send_reply;
Else
                        discard_req;
End if
```

In case the receiving node belongs to class D which is composed of notorious nodes, the request packet is discarded so that a notorious node will not be part of route.

In case, if the node receiving request packet is intermediate node and it has not received the same request packet earlier then it will increment the *counter* of either class to whom it belongs. After incrementing the field, it will further broadcast the packet. If its history is equal to or more than threshold history it will not further broadcast the packet. If this node again receives the request packet with same req_id, it will simply discard it to avoid network congestion. This discarding of request packet with same id will help in selecting multiple disjoint routes. Now if intermediate node has a route in its routing table it will send reply to source with available route. The reply packet also contains three fields for classes. When a route reaches to source from intermediate node or from destination, proposed protocol will start to transmit data which comes first and when other route reply comes either from destination or from intermediate node proposed protocol will compare these routes by their ratios and switch to best one.

If node receiving request packet is destination, then all request packets reaching to it are making disjoint route. Here HBSR protocol will select best route based on the movement history. As each request packet shows how many nodes belong to each class along with total hop count. The destination will compute the ratio of all class nodes with total hop count of the route of each request packet. First the ratio of third class is compared for all available routes. A route with less third class ratio is more stable than a route with more third class ratio. So a route with less ratio will be selected. Here if two or more routes have same third class ratio, then second class nodes ratio of all packets will be compared. A route with less second class ratio will be selected. If it is also same, then first class ratios will be compared for decision-making. Now destination will send two disjoint best routes to the source. The reply packet has three fields for classes to show total no of nodes in each class.

4. Result

Consider a MANETs network of 33 nodes given in figure 1

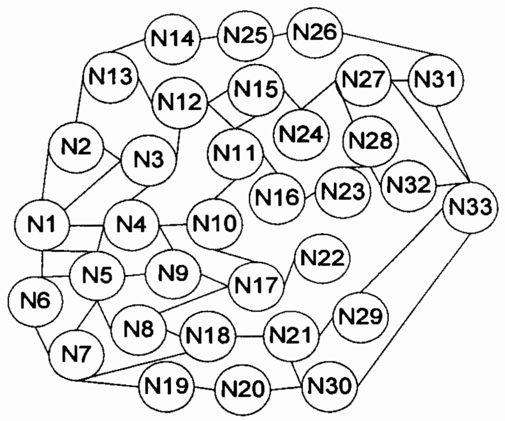

Figure 1: A MANETs network of 33 nodes

Suppose N1 want to transmit data to N33. It will send request packet to all its neighbors to find the route to N33. The request packet moves from one node to other in order to reach the destination N33. A node will further forward the request packet if it has not received this request packet already. In case it has received it already it will discard the request packet. If we follow the path of request packet to destination, the path translation is as follows

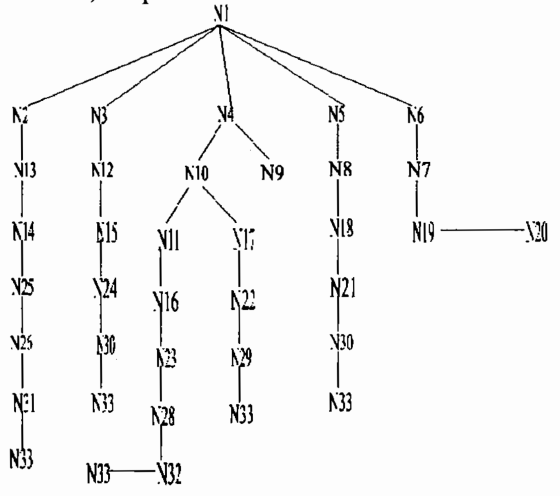

Figure 2: Path translation from N 1 to N33

The possible paths from N1 to N33 are

ROUTE #1 : N1→N2 →N13 →N25 →N26 →N31 →N33
ROUTE # 2 : N1→N3 →N12 →N15 →N24 →N27 →N33
ROUTE #3 : N1→N4 →N10 →N11 →N16 →N23 →N28 →N32 →N33
ROUTE #4 : N1→N4 →N10 →N17 →N22 →N29 →N33
ROUTE #5 : N1→N5 →N8 →N18 →N21 →N30 →N33
ROUTE #6 : N1 →N6 →N7 →N19 →N20 ×
ROUTE #7: N1 →N4 →N9 ×

Let suppose threshold history is 6 then we define the classes A, B, C, D as follows
Class A = all nodes that have history 0 or 1
Class B = all nodes that have history 2 or 3
Class C = all nodes that have history 4 or 5
Class D = Notorious nodes = that have history 6 or above
Further suppose
Class A = N6, N7, N26, N33, N21
Class B = N1, N4, N9, N11, N15, N13, N14, N25, N28, N20, N12, N18
Class C = N2, N3, N8, N30, N5, N31, N24, N10, N23, N32, N17, N29, N22, N19, N16, N27

Here in this example, route 6 and 7 are not complete because all the neighbors of N9 in route 7 and N20 in route 6 have received the request packet already from other neighbors so they will discard the request packet to maintain the disjoint paths. The class C ratio of Route #1 is 2/8, route #2 is 3/7, route#3 is 4/9, route#4 is 4/7, and route #5 is 3/7. On this comparison of class C ratio, route #1 is selected as it has least class C ratio. Whereas route #2 and route #5 have same class c ratio i.e. 3/7 so their class B ratio will be compared. Class B ratio of route 2 is 3/7 and route 5 is 2/7. Thus route 5 will be selected. Now destination node N33 will reply N1 with two routes, route 1 and route 5.

This is example where intermediate nodes have no routes to the destination. If there is some node let say N10, has a route to the destination N33, it will reply the N1 with that route along with all class ratios. N1 on receiving this reply will start transmitting data using this route. Later on, when N1 receives the reply packets from N33, it will compare the ratios of three reply packets and select two best routes.

5. Conclusion

Mobile Ad-hoc Networks is the emerging field in computer networks. MANETs are difficult to handle because of their some basic features and they are widely used due to their applications. Routing issues, security issues and QoS issues in the MANETs are also discussed.

As the primary focus was on the path stability, that is the most important issue in any of the QoS model. Some common protocols designed for path stability are also discussed. These protocols are the extensions of reactive, pro-active and hybrid approaches. Some focus on the fast link repair procedure in case of link breakage and some on the selection of stable behavior nodes in route selection in order to minimize the chances of link breakage

In the last section, we present problem that for achieving any QoS model route stability is the main factor. If we succeeded in achieving route stability we can achieve QoS routing. Our proposed protocol named "history based stable route protocol (HBSR) "will provide route stability. By selecting stable routes, following benefits will be achieved.

i) Reduces consumption of bandwidth for control traffic
ii) Reduces delay during transmission
iii) Provides multiple disjoint path
Provides benefits of both reactive and proactive protocols

6. References

[1] Alejandro Quintero, Samuel Pierre, Benjamin Macabeo, A routing protocol based on node density for ad hoc networks, Ad Hoc Networks 2 (2004) 335–349.
[2] Charles Perkins, Ad-Hoc On Demand Distance Vector (AODV) Routing: draft-ietf-manet-aodv-00.txt, 20 November 1997.
[3] Youngki Hwang and Pramod Varshney, Fellow, IEEE, An Adaptive QoS Routing Protocol with Dispersity for Ad-hoc Networks, Proceedings of the 36th Hawaii International Conference on System Sciences: (HICSS'03) 0-7695-1874-5/03 $17.00 © 2002 IEEE
[4] Chun-Yuah Chiu, Eric Hsiao-kuang Wu, and Gen-Huey Chen, Stability Aware Cluster Routing Protocol for mobile Ad-Hoc Networks., Proceedings of the Ninth International Conference on Parallel and Distributed Systems (ICPADS'02) 1521-9097/02 © 2002 IEEE.
[5] Michael Gerharz, Christian de Waal, Peter Martini and Paul James, Strategies for Finding Stable Paths in Mobile Wireless Ad Hoc Networks, Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks (LCN'03) 0742-1303/03 © 2003 IEEE.
[6] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario-based performance analysis of routing protocols for mobile ad-hoc networks. In 5th Annual ACM/IEEE International Conference on Mobile

Computing and Networking (Mobicom), pages 195.206, Seattle Washington, August 1999.

[7] Robert Radziwilowicz, a presentation "Quality of Service in Ad Hoc Networking" SITE, University of Ottawa, 05 November 2004.

[8] Harpreet S. Arora, *Towards Achieving QoS Guarantees in Mobile Ad- Hoc Networks*, M.S. thesis, Drexel University, October 2003