Performance Evaluation of DSR and DSDV

Dissertation submitted to Jawaharlal Nehru University, in partial fulfillments of the requirements for award of the degree of

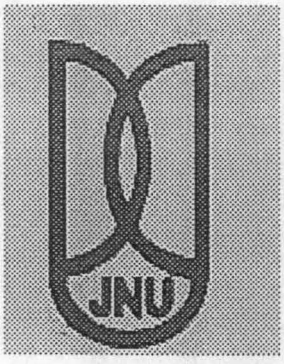
Master of Technology

In

Computer Science and Technology

By

ANIL KUMAR



SCHOOL OF COMPUTER & SYSTEMS SCIENCES

JAWARLAL NEHRU UNIVERSITY

NEW DELHI -110067

January 2004

my Copy

Performance Evaluation of DSR and DSDV

Dissertation submitted to Jawaharlal Nehru University, in partial fulfillments of the requirements for award of the degree of

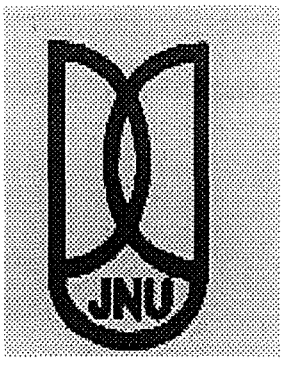
Master of Technology

In

Computer Science and Technology

By

ANIL KUMAR



SCHOOL OF COMPUTER & SYSTEMS SCIENCES

JAWARLAL NEHRU UNIVERSITY

NEW DELHI -110067

January 2004

CERTIFICATE

This is to certify that the project entitled "**Performance evaluation of DSR and DSDV**" being submitted by Anil Kumar to the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, in partial fulfillment of the requirements for the award of the degree of Master of Technology in Computer Science & Technology, in a bonafide work carried out by me under the guidance and supervision of Dr. D. K. Lobiyal.

The matter embodied in the dissertation has not been submitted for the award of any other degree or diploma.

Anil Kumar

Dr. D. K. Lobiyal
Asst. Professor, SC&SS,
JNU, New Delhi-67

Prof. Karmeshu
Dean, SC&SS,
JNU, New Delhi-67

Dedicated to

My beloved

Parents

ACKNOWLEDGEMENTS

I would like to pay obeisance of the feet of my parents for their blessings that are always with me in all my aspirations including my academics.

I would like to sincerely thank my supervisor Dr D. K. Lobiyal, School of Computer And System Sciences, Jawaharlal Nehru University for the help, encouragement and support extended by him in successful completion of this project. His ideas were innovative and the valuable discussions we had were very much helpful in keeping the project on the right track.

I would like to record my sincere thanks to the dean Prof. Karmeshu for providing the necessary facilities. I extend my sincere gratitude to my lab mates for their continuous academic as well as morale support through out my dissertation work. I acknowledge the efforts and knowledge put by the development team of project at team UC Berkeley, for developing the network simulator and making its source code available on the Internet as a freeware.

Last, but not the least, I take this opportunity to thank all the faculty, members and friends for their, help and encouragement during the course of the project.

Anil Kumar

ABSTRACT

In this dissertation work simulation study of routing protocols, dynamic source routing (DSR) and destination sequenced distance vector (DSDV) have been done by using the network simulator, NS2 developed at UC Berkeley. The network simulator NS2 provides this simulation environment for the ad hoc networks.

These two algorithms are run under the simulator and its performances in terms of throughput, routing overhead, power consumption have been studied for varying the different parameters such as node speed, pause time, network size, and number of traffic sources (pattern).

Different types of scenario files are generated to evaluate the performance of the routing protocols by using the scenario files generator in the network simulator .In the implementation, TCL script has been used to configure the network and to simulate it .A routine in C has been written to extract this simulation results from the output file generated by network simulator result of simulations have been by shown by the graphs drawn in MS-excel.

CONTENTS

CHAPTER-1

INTRODUCTION

1.1 BACKGROUND

Wireless communication between mobile users is becoming more popular than ever before. This is due to recent technological advances in lap top computers and wireless data communication devices, such as wireless modems and wireless LANs [9].

There are two distinct approaches for enabling wireless communication between two hosts. The first approach is to let the existing cellular network infrastructure carry data as well as voice. The second approach is to form an ad-hoc network among all users wanting to communicate with each other. This means that all users participating in the ad-hoc network must be willing to forward data packets to make sure that the packets are delivered from source to destination. This form of networking is limited in range by the individual nodes transmission ranges and is typically smaller compared to the range of cellular systems. This does not mean that the cellular approach is better than the ad-hoc approach.

Ad-hoc networks do not rely on any pre-established infrastructure and can therefore be deployed in places with no infrastructure. Because nodes are forwarding packets for each other, some sort of routing protocol is necessary to make the routing decisions. As the cost of Wireless access drops, wireless communications could replace wired in many settings. The advantage of wireless is the ability to transmit data among the users in a common area while remaining mobile. Mobile nodes such as notebook computers, featuring powerful CPUs large main memories, hundreds of megabytes of disk space are

now easily affordable and becoming quite common in every day business and personal life. At the same time, network connectivity options for use with mobile hosts have increased dramatically, including support for a growing number of wireless networking products based on radio and infrared [10].

1.2 Mobile ad-hoc network

A mobile ad-hoc network is a collection of mobile nodes with no pre established infrastructure, forming a temporary network. Each of the nodes has a wireless interface and communicates with each other over radio or infrared [5].

Laptop computers and personal digital assistants that communicate directly with each other are some examples of nodes in an ad-hoc network. Nodes in the ad-hoc network are often mobile, but can also consist of stationary nodes, such as access points to the internet.

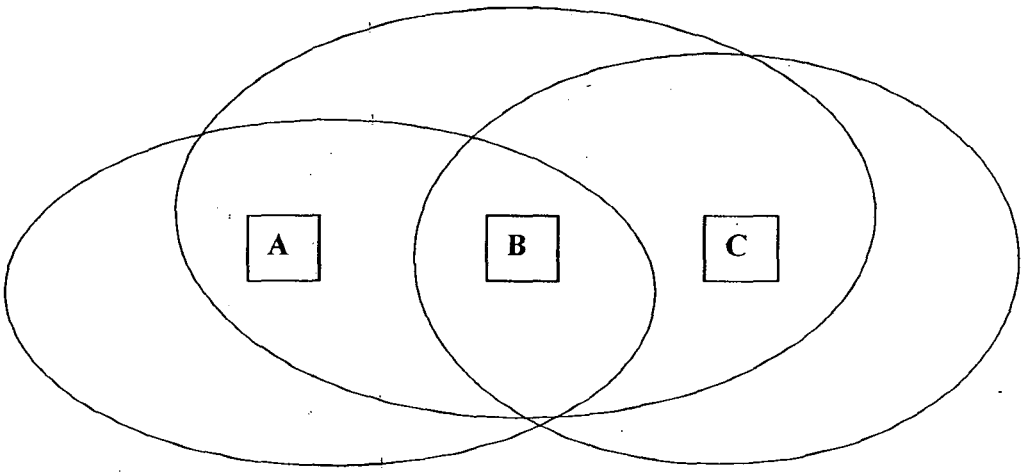


Figure 1.1: A simple ad-hoc network of three wireless mobile hosts

If only two hosts, located closely together, are involved in the ad hoc network, no real routing protocol or routing decisions are necessary. In many ad hoc networks, though,

two hosts that want to communicate may not be within wireless transmission range of each other, but could communicate if other hosts between them also participating in the ad hoc network are willing to forward packets for them.

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

Indeed, the routing problem in a real ad hoc network may be more complicated than this example suggests, due to the inherent non-uniform propagation characteristics of wireless transmissions and due to the possibility that any or all of the hosts involved may move at any time.

An ad-hoc network uses no centralized administration .This is to be sure that the network won't collapse just because one of the mobile nodes moves out of transmitter range of the others. Nodes should be able to enter/leave the network as they wish. Because of the limited transmitter range of the nodes, multiple hops may be needed to reach other nodes. Every node wishing to participate in an ad-hoc network must be willing to forward packets for other nodes. Thus every node acts both as a host and as a router. A node can be viewed as an abstract entity consisting of a router and a set of affiliated mobile hosts (fig 1.2).A router is an entity, which among other things runs a routing protocol. A mobile host is simply an IP-addressable host/entity in the traditional sense [8].

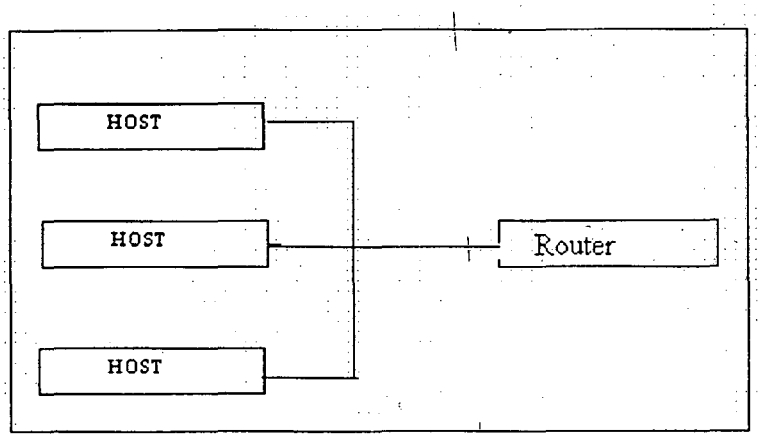


Figure1.2: Block diagram of a mobile acting as host and router

1.3 Usage

There is no clear picture of what these kinds of networks will be used for. The suggestions vary from document sharing at conferences to infrastructure enhancements and military applications. In areas where no infrastructure such as the internet is available an ad-hoc network could be used by a group of wireless mobile hosts. This can be the case in areas where a network infrastructure may be undesirable due to reasons such as cost or convenience. Examples of such situations include disaster recovery personnel or military troops in cases where the normal infrastructure is either unavailable or destroyed.

Other examples include business associates wishing to share files in an airport terminal, or a class of students needing to interact during a lecture. If each mobile host wishing to communicate is equipped with a wireless local area network interface, the group of mobile hosts may form an ad-hoc network.

1.4 Characteristics of Ad hoc Networks

Ad-hoc networks are often characterized by a dynamic topology due to the fact that the nodes change their physical location by moving around. This favors routing protocols

that dynamically discover routes over conventional routing algorithms like distance vector and link state.

Another characteristic is that a host/node has very limited CPU capacity, storage capacity, battery power and bandwidth. This means that the power usage must be limited thus leading to a limited transmitter range.

The access media, the radio environment, also has special characteristics that must be considered when designing protocols for ad-hoc networks. One example of this may be unidirectional links. These links arise when for example two nodes have different strength on their transmitters, allowing only one of the hosts to hear the other, but can also arise from disturbances from the surroundings.

The characteristics of mobile ad-hoc network are summarized as:

- There is no centralized administration or standard support services.
- Posse's dynamic topology due to the fact that nodes are free to move arbitrary.
- Uses multi-hop routing in which nodes that are not in each other's transmission range can communicate through the intermediate nodes.
- These networks may consist of both unidirectional and bi-directional links.
- These networks are more prone to physical security threats than are fixed cable networks.

1.5 Applications of Ad hoc Networks

There are several areas where ad-hoc networks are useful:

- Personal area networking – This type of networking includes cell phone, laptops, earphone, and wrist watch.
- Military environments- This type of networking is also used for soldiers, tanks, planes.

- Civilian environments- This type of networking is also used for taxicab network, meeting rooms, sports stadiums, boats, small aircraft.
- Emergency operations- This type of networking is also used for search and rescue, policing and fire fighting.

1.6 Routing in Ad hoc Networks

Because of the fact that it may be necessary to hop several hops (multiple hops) before a packet reaches the destination, a routing protocol is needed. The routing protocol has two main functions, selection of routes for various source-destination pairs and the delivery of messages to their correct destination. The second function is conceptually straight forward using a variety of protocols and data structures (routing tables) [14].

1.7 Routing protocols in Ad-hoc Networks

❖ **Dynamic source routing:**

Dynamic source routing algorithm belongs to the class of reactive protocols and slows nodes to dynamically discover a route across multiple network hops to any destination. Source routing means that each packet in its header carries the complete ordered list of nodes through which the packet must pass [6].

In route discovery phase the source code that needs a route to some destination node requests a route by broadcasting a Route Request (RREQ) packet. Every node receiving this RREQ searches through its route cache for a route to the requested destinations. DSR stores all known routes in its route cache. If no route is found, it forwards the RREQ further and adds its own address to the recorded hop sequence. This request propagates through the network until either the destination or a node with a route to the destination is reached. When this happen a route reply (RREP) is unicasted back to the originator. This RREP packet contains the sequence of network hops through which it may reach the target.

Route maintenance is the mechanism in which a source node S detects if network topology has changed, so that it can no longer use its route to some destination node D. when route maintenance detects a problem with a route in use , a route error packet is sent back to the source node. When this error packet is received, the hop in error is removed from this host's route cache, and all routes that contain this hop are truncated at this point.

❖ **Destination sequenced distance vector (DSDV):**

DSDV is a proactive table driven hop by hop distance vector routing protocol that in each node has a routing table that for all reachable destinations stores the next hop and number of hop for that destination. Each node periodically broadcast routing updates [3].

To guarantee loop-freedom DSDV uses a sequences numbers to tag each route. The sequence number shows the freshness of a route and routes with higher sequence numbers are favorable. A route R is considered more favorable than R' if R has a greater sequence number or, if the routes have the same sequence number or, if the routes have the same sequence number but R has low hop-count. The sequence number is increased when a node A detects that a route to a destination D has broken. So the next time node A advertises its route, it will advertise the route to D with an infinite hop-count and sequence number that is larger than before.

DSDV basically is distance vector with small adjustments to make it better suited for ad-hoc networks. These adjustments consist of triggered updates that will take care of topology changes in the time between broadcasts. To reduce the amount of information in these packets there are two types of update messages defined: full and incomplete dump. The full dump carries all available routing information and the incremental dump that only carries the information that has changed since the last dump.

❖ Ad-hoc on demand distance Vector Routing (AODV)

AODV is pure on demand, loop free routing protocol obtained by building on the DSDV algorithm. AODV only requests a route when needed and does not require nodes to maintain route to destinations that are not actively used in communication. The algorithm uses different messages like route request (RREQ) and hello message (RREP) to discover and maintain routes [2].

When source node needs a route to some destination node, and does not already have a valid route to that destination, it initiates route discovery process. It broadcasts a route request (RREQ) packets to its neighbors, which then forward the request to their neighbors, and so on, until either the destination or an intermediate node with a fresh enough route to the destination is located. Then a route is made available by unicasting a RREP back to the source. In the route maintenance process when a node detects a route to a neighbor that no longer is valid, it will remove the routing entry and send a link failure message, a triggered route reply message to the neighbors that are actively using the route, informing them that this route no longer is valid. The nodes that receive this message will repeat this procedure. The message will eventually be received by the affected sources that can chose to either stop sending data or requesting a new route by sending a new route request RREQ.

❖ Temporally Ordered Routing Algorithm (TORA)

Temporally Ordered Routing Algorithm is a highly adaptive, loop free, distributed routing algorithm based on the concept of link reversal. It is source-initiated and provides multiple routes for any desired source destination pair. The key design concept of TORA is the localization of control messages to a very small set of nodes near the occurrence of a topological change. TORA performs three basic functions route creation, route maintenance, and route erasure.

During the route creation and route maintenance phases, nodes use a "height" metric to establish a directed acyclic graph (DAG) rooted at the destination. Then the links are assigned a direction, either upstream or down stream based on the relative height metric of neighboring nodes. If a route is broken with node mobility, route maintenance is necessary to re-establish the DAG rooted at the same destination. Up on failure of the last down stream link, a node generates a new reference level, which results in the propagation of that reference level by neighboring nodes, effectively coordinating a structured reaction to the link failure. Links are reversed to reflect the change in adapting to the new reference level.

The "height" metric in TORA is dependent on the logical time of link failure. The "height" metric consists of five elements, namely logical time of a link failure, the unique ID of the node that defined the new reference level, a reflection indicator bit, a propagation ordering parameter, and the unique ID of the node. TORA's route erasure phase involves flooding a broadcast clear packet (CLR) throughout the network to erase invalid routes

1.8 Problem definition:

Several factors affect the performance of any protocol operating in ad-hoc network. For example, node mobility may cause link failures. The primary objective of this project is to evaluate the affects of the following parameters on DSR and DSDV:

- Node speed.
- Pause time.
- Traffic pattern.
- Network size.

We determine the impacts of these parameters on the following performance metrics:

- Average throughput.
- Average routing overhead.
- Power consumption.

So the goal of this project is to:

- Get a general understanding of ad-hoc networks.
- Generate a simulation environment that could be used for further studies.
- Implement the proposed routing protocols for ad-hoc networks.
- Analyze the protocols theoretically and through simulation.
- Recommend protocols for specific network scenario.

CHAPTER-2

DYNAMIC SOURCE ROUTING ALGORITHM

2.1 INTRODUCTION

The dynamic source routing algorithm (DSR) is a simple and efficient routing protocol designed specifically for use in multi hop wireless ad hoc networks of mobile nodes. DSR allows the network to be completely self-organizing and self-configuring, with out the need for any existing infrastructure or administration. Nodes in the network cooperate to forward packets to each other to allow communication over the multiple hops, which are not in wireless transmission range of each other. As the nodes in the network can move anywhere, can join or leave the network at any time and as wireless conditions like sources of interference change, all routing is automatically determined and is maintained by the DSR protocol. As the number of sequence of intermediate hops needed to reach any destination may change at any time, the resulting network is highly dynamic in nature and is rapidly changing [6].

In DSR algorithm every data packet sent contains in its header the complete ordered list of nodes through which packet will pass. This type of routing is known as source routing. The DSR protocol enables the nodes to discover a source route dynamically to any destination across multiple hops or across multiple network hops. Hence, the name dynamic source routing algorithm. This algorithm allows the packet routing to be loop-free and avoids the need for up-to-date routing information in the intermediate nodes through which the packet is being forwarded. By appending the source route in the packet header, the other nodes forwarding or overhearing any of these data packets may also easily cache this routing information for future use.

The protocol is composed of the two mechanisms Route Discovery and Maintenance, which allow nodes to discover and maintain source routes to arbitrary destination in the ad hoc network. All aspects of the protocol operate entirely on-demand, allowing the routing packet overhead of DSR to scale automatically to only those nodes that need to react to the changes in the routes currently in use.

Route Discovery is the mechanism by which a node A willing to send a packet to a destination node **G**, obtains the path to G, Routing discovery is used only when A wants to send a packet to node G but does not know a path to **G.**

Route Maintenance **is** the mechanism through which a node A is able to detect that it can no longer user its route to G because a link along the route no longer exists due to a change in broken, A can attempt to use any other route it happens to know to G, or can invoke Route Discovery again to find a new route for subsequent packets, Route Maintenance for this route is used only when A is actually sending packets to **G.**

In DSR algorithms Routing Discovery and Route Maintenance each operate entirely on demand. DSR algorithm requires no periodic packets of any kind at any level within the network. For example DSR algorithms does not use any periodic routing advertisement, like status sensing, or neighbor detection packets, and does not relay on this functions from any under lying protocol in the network. This entirely on-demand behavior and lack of periodic activity allows the number of overhead packets cause by DSR to scale all the way down to zero, when all nodes are approximately stationary with respect to each other and all routes needed for current communication have already been discovered. As nodes begin to move more or as communication have already been discovered. As nodes begin to move more or as communication patterns change, the routing packets overhead of DSR automatically scales to only that needed to track the routes currently in use. Network topology changes not affecting routes currently in use are ignored and do not cause reaction from the protocol. In respect to a single Route Discovery (as well as through routing information from other packets overheard), a node may learn and cache multiple routes to any destination. This allows the reaction to routing changes to be much more

rapid, since a node with multiple routes to a destination can try another cache route if the node one it has been using fails. This caching of multiple routes also avoids the overhead of needing to perform a new Route Discovery each time a route in use breaks. The operation of both Route Discovery and Route Maintenance in DSR are designed to allow unidirectional links and asymmetric routes to be easily supported. In wireless networks, it is possible that a link between two nodes may not work equally well in both directions, due to differing antenna or propagation patterns or source of interference. DSR allows such unidirectional links to be used when necessary, improving overall performance and network connectivity in the system.

2.2 Assumptions

All the hosts willing to communicate with others host in the ad hoc network, by using the DSR algorithms are assumed to participate fully in the protocols of the network. Each host participating in the network should also be willing to forward packets to the other nodes in the network.

The number of hops required for a packet to each from one host located at the one extreme edge of the network to the opposition extreme is known as the diameter of the network.

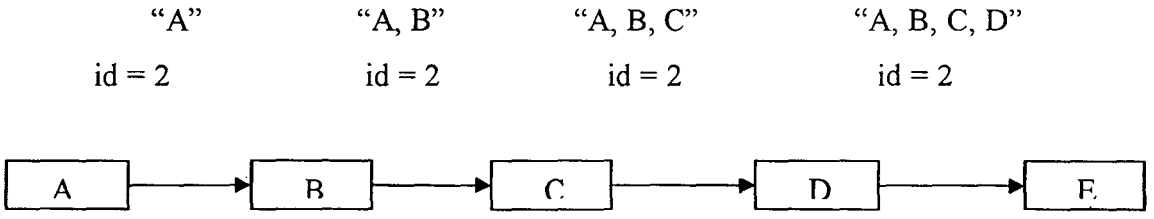
Hosts within the ad hoc network may move at any time without notice but we assume that the speed with which hosts move is moderate with respect to the packet transmission latency and wireless transmission range of the particular underlying network hardware in use. We assume that hosts do not continuously move so rapidly
As to make the flooding of each packet the only possible routing protocol.

2.3 Protocol Overview

2.3.1 Basic Route Discovery

When a source node originates a packet to destination node, the source node places in the header of the packet a source route giving the sequence of hops that the packet has to follow in its way to the destination .The sender will obtain a suitable source route by searching . Its Route Cache of routes previously learned , but if no route is found in its cache for the intended destination then it initiates a route discovery to find the route to the destination dynamically . The source node is called the initiator and the destination node is called as the target of the route discovery.

For example let us consider the following scenario node A wants to initiate a route discovery to node E. The route discovery initiated by node A in this scenario will be as follows.



To initiate the route discovery the node A transmits a *route request* broadcast packet to all the nodes with in its wireless range .Every route request identifies the initiator and target of the route request discovery, and also contains a unique identification(in this example it is 2) determined by the initiator of the request.

Each route request also contains a record listing the address of each intermediate node through which the route request packet has been propagated. The route request initially lists only the node A. When another node receives the route request , if it is the target of

route discovery , it also gives a copy of the accumulated route record from the route request .When the initiator receives the route reply it caches this route in its route cache for use in sending subsequent packets to this destination.

If this node receiving the route request has recently seen another route request message from this initiator bearing the same request identification and target , or if this node's own address is already listed in the route record in the route request, this node discards the request. This node appends its own address to the route record in the route request and propagates it by transmitting it as a local broadcast packet (with the same request identification).

When initiating the route discovery, the sending node saves a copy of the original packet in a local buffer called the send buffer. The send buffer contains a copy of each packet in a local buffer called the send buffer. The send buffer contains a copy of each packet that cannot be transmitted by this node because it does not yet have a source route to the packet's destination. Each packet in the send buffer is logically associated with the time it was placed in to the buffer. The packet is discarded if the packet still resides in the send buffer for that time period.

2.3.2 Route Maintenance

When originating or forwarding a packet using a source route, each node transmitting the packet is responsible for confirming that the packet should be retransmitted until this confirmation of receipt is received. This confirmation can be obtained as a MAC layer acknowledgement or as a passive acknowledgement. If none of the two confirmations are received the node transmitting the packet can explicitly request a DSR- specific software acknowledgement can be returned by the next hop. This software acknowledgement can normally be transmitted. Directly to the sending node. If the link between the nodes is unidirectional then the software acknowledgement may take a different multi hop path.

If no receipt information is received after the packet has been retransmitted the maximum no of attempts by the same hop. This node should return a route error packet to the original sender of the packet, identifying the link on which the packet could not be forwarded.

2.4 Salient Features of the Algorithm

❖ Caching of Overhead Routing Information

A node forwarding or overhearing any packet may add the routing information from that packet to its own Route Cache .In particular, the source route used in a data packet, any node may cache all the accumulated route record in a route request , or on the route being returned in route reply .Routing information from any of these packets received can be cached , whether the packet was addressed to this node , sent to a broad cast MAC address, or received while the node's network interface is in promiscuous mode.

❖ Replaying to a Route Requests using Cached Routes

A node receiving a route request for which it is not the target searches in its route cache for a route to the target of the target of the request. If found, the node generally returns a route reply to the initiator itself rather than forwarding the route request. In the route reply, this node sets the route record to list the sequence of hops over which this copy of the route request was forwarded to it, concatenated with the source route to this target obtained from its own route cache .However, before transmitting a route reply packet that was generated using information from its route cache in this way, a node must verify that the resulting route being returned in the route reply , after this concatenation, contains no duplicate nodes listed in the route record.

❖ **Preventing Route Reply Storms**

The ability for nodes to reply to a route request based on information in their route caches could result in a possible route reply "storm" in some cases. In particular, if a node broadcasts a route request for a target node for which the node's neighbors have a route in their route caches, each neighbor may attempt to send a route reply, thereby wasting bandwidth and possibly increasing the number of network collisions in the area.

Normally, these nodes would all attempt to reply from their own route caches and would all send their route replies at about the same time, since they all received the broadcast route request at about the same time. Such simultaneous replies from different nodes all receiving the route request may create packet collisions among some or all of these replies and may cause local congestion in the wireless network. In addition, it will often be the case that the different replies will indicate routes of different lengths.

❖ **Route Request Hop limits**

Each Route Request message contains a " hop limit" that may be used to limit the number of intermediate nodes allowed to forward that copy of the route request .This hop limit is implemented using the Time-to-Live(TTL) field in the IP header of the packet carrying the route request. As the request is forwarded, this limit is decremented, and the request packet is discarded if the limit reaches zero before finding the target. This route request hop limit can be used to control the spreading of a route request during a route discovery attempt. For example, a node may send its first route request attempt for some target node using a hop limit of 1 , such that any node receiving the initial transmission of the route request will not forward the request to other nodes by rebroadcast it. This form of route request is called a "non-propagating "route request. It provides an inexpensive method for determining if the target is currently a neighbor of the initiator or if a neighbor node has a route to the target cached.

❖ **Automatic Route Shortening**

Source routes in use may be automatically shortened if one or more intermediate hops in the route become no longer necessary, This mechanism of automatically shortening routes in use is somewhat similar to the use of passive acknowledgements in particular, if a node is able to overhear a packet carrying a source route (e.g., by operating its network interface in promiscuous receive mode), then this node examines the unused portion of that source route. If this node is not the intended next hop for the packet but is named in the later unused portion of the packet's source route, then it can infer that the intermediate nodes before itself in the source route are no longer needed in the route.

❖ **Increased Spreading of Route Error Messages**

When a source node receive a route error for a data packet that it originated, this source node propagates this route error to its neighbors by piggybacking it on its next route request .In this way, stale information in the caches of nodes around this source node will not generate route replies that contain the same invalid link for which this source node received the route error .On the route request packet initiating this route discovery , node A piggybacks a copy of this route error , ensuring that the route error spreads well to other nodes , and guaranteeing that any route reply that it receives (including those from other node's route caches) in response to this route request does not contain a route that assumes the existence of this broken link.

2.5 Conceptual Data Structures

DSR algorithm uses the following data structures for transferring data across a mobile ad hoc network.

❖ Route Cache

All routing information needed by a node participating in an ad hoc network using DSR is stored in that node's route cache. Each node in the network maintains its own route cache .A node information to its route cache as it learns of new links between nodes in the ad hoc network , for example , a node may learn of new links when it receives a packet carrying either a route reply or a DSR routing header .Likewise, a node removes information from its route cache as it learns that existing links in the ad hoc network have broken for example, a node may learn of a broken link when it receives a packet carrying a route error or through the link-layer retransmission mechanism reporting a failure in forwarding a packet to its next-hop destination .It is possible to interface a DSR network with other network with other networks , external to this DSR network .Such external network may be the internet , or may be other ad hoc networks that are treated as external networks in order to improve scalability .This minimal set of requirements an features involve the first hop External(F) and Last Hop External(L) bits in a source route option and a route reply option in a packet's DSR header. These requirements also include the addition of an external flag bit tagging each node in the route cache, copied from the First Hop External(F) and Last Hop External(L) bits in the source route option or route reply option from which the link to this node was learned .The route cache should support storing more than one route to each destination .In searching the route cache for a route to some destination node, the route cache is indexed by destination node address. An implementation of a route cache may provide a fixed capacity for the cache or the cache size can be a variable.

❖ Route Request Table

The route request table records information about route requests that have been recently originated or forwarded by this node .The table is indexed by IP address .The route request table on a node records the following information about nodes to which this node has initiated a route request.

- The time that this node lost originated a Route request for that target node
- The number of consecutive route request initiated for this target since receiving a valid route reply giving a route to that target node.
- The remaining amount of time before which this node may next attempt at a route discovery for that target node
- The time_to_live (TTL) field used in IP header of last route request initiated by this node for that target node.

In addition the route request table on a node also records the following information about initiator nodes from which this node has received a route request.

A FIFO cache of size REQUEST_TABLE_IDS entries containing the identification value and target address from the most recent route requests received by this node from that initiator node

Node should use an LRU policy to manage the entries in their route request table. The number of identification values to retain in each route request table entry, REQUEST_TABLE_IDS ,must not be unlimited, since, in the worst case , when node crashes and reboots, the first REQUEST_TABLE_IDS route discoveries it initiates after rebooting could appear to be duplicates to the other nodes in the network. In addition, a node should base it's initial identification value, used for route discoveries after rebooting, on a battery backed up clock or other persistence memory device, in order to help avoid any possible such delay in successfully discovering new routs after rebooting if no such source of initial identification values is available, a node should base it's initial identification value after rebooting on a random number.

❖ **Send Buffer**

The sand Buffer of a node implementing DSR is a queue of packets that can not be sent by that node because it does not yet have a source route to each such packet's destination.

Each packet in the send buffer is logically associated with the time that it was placed into the buffer, and should be removed from the send buffer and silently discarded SEND_BUFFER_TIMEOUT seconds after initially being placed in the buffer.
if necessary, a FIFO strategy can be used to evict packets before they time out to prevent the buffer from overflowing. A route discovery should be initiated as often as possible for the destination address of any packets residing in the send buffer.

❖ **Retransmission Buffer**

The retransmission Buffer of a node implementing DSR is a queue of packets send by this node that are awaiting the receipt of an acknowledgment from the next hop in the source route. For each packets retransmission buffer, a node maintains

- A count of the number of retransmissions and
- The time of the last retransmission

Packets are removed from the retransmissions buffer when an acknowledgement is received or when the number of retransmissions exceed DSR_MAXRXTSHIFT. In the later case, the removal of packet from the retransmission buffer should result in a route error being returned to the original source of the packets.

CHAPTER-3

DESTINATION-SEQUENCED DISTENCE VECTOR PROTOCOL

3.1 Protocol Overview

The destination-Sequenced Distance vector (DSDV) Routing algorithm is based on the idea of the Distributed Bellman-ford (DBF) Routing Algorithm with certain improvements. Its memory requirement is very moderate and it guarantees loop-free paths all instants [3].

In DBF Routing Algorithm, every node I maintains, for each destination x, a set of distances $\{d^{xij}\}$ Where j ranges over the neighbors of i. Node I treats neighbor k as a next-hope for a packet destined for x if $d^{x\,ik}$ equals minj $\{d^{x}ij\}$. The succession of next hops chosen in this manner lead to x along the shortest path. In order to keep the distances estimates up-to-date, each node monitors the cost of its outgoing links and periodically broadcasts, to each one its neighbors, its current estimate of the shortest distance to every other node in the network.

In DSDV Routing Algorithm, Packets are transmitted between the stations of the network by using routing tables, which are stored at each station of the network.. Each routing table, at each of the station, lists all available destinations, and the number of hops to each. Each route table entry tagged with a sequence number, which is originated, by the destination station. To maintain the consistency of routing tables in a dynamically varying topology, each station periodically transmits updates, and transmits updates immediately when significant new information is available.

Since we do not assume that the mobile hosts are maintaining any sort of time synchronization, we also make no assumption about his phase relationship of the update periods between the mobile hosts. These packets indicate which stations are accessible from each station and the number of hops necessary to reach these accessible stations, as is often done in distance-vector routing algorithms. The packets may be transmitted containing either layer 2 (MAC) address of layer 3 (network) addresses [13].

Routing information is advertised by broadcasting or multicasting the packets, which a transmitted periodically and incrementally as topologically as changes, are detected for instance, when stations move within the network. Data is also kept about the length of time between arrival of the first and arrival of the best route for each particular destination. Based on this data, a decision may be made to delay advertising routes, which a are about to change soon, thus damping fluctuations of the route tables. The advertisement of routes, which may not have stabilized, yet is delayed in order to reduce the number of rebroadcasts of possible route entries that normally arrive with same sequence number.

The DSDV protocol requires each mobile station to advertise, to each of its current neighbors, its own routing table (for instance, by broadcasting it entries). The entries in this list may change fairly dynamically over time, so the advertisement must be made often enough to ensure that every mobile computer can almost always locate every other mobile computer of the collection. In addition, each mobile computer agrees to relay data packets to other computers upon request. This agreement places a premium on the ability to determine the shortest number of hops for a route to a destination. We would like to avoid unnecessary disturbing mobile hosts if they are in sleep mode. In this way a mobile computer may exchange data with any other mobile computer in the group even if the target of the data is not within range of direct communication.

3.2 Routing Table Management

All the computers interoperating to create data paths between themselves broadcasts the necessary data periodically say once every few seconds. In a wireless medium, it is important to keep in mind, that broadcasts are limited in range by the physical characteristics of the medium. This is different than the situation with wired media, which usually have a much more well-defined range of reception. The data broadcast by each mobile computer will contain its new sequence number and the following information for each new route.

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

The transmitted routing tables will also contain the hardware address, and (if appropriate) the network address, of the mobile computer transmitting them, within the headers of the packet. The routing table will also include a sequence number created by the transmitter. Routes with more recent sequence numbers are always preferred as the basis for making forwarding decision's but not necessarily advertised. Of the paths with the same sequence number, those with the smaller metric will be used. By the natural way in which the routing tables are propagated, the sequence number is sent to all mobile computers, which may each decide to maintain a routing entry for that originating mobile computer.

Routes received in broadcasts are also advertised by the receiver when it subsequently broadcasts its routing information, the receiver adds and increment to the metric before advertising the route, since incoming packets will require one more hop to reach the destination (namely, the hop from the transmitter to the receiver).

One of the mot important parameters to be chosen is the time between broadcasting the routing information packets. However, when a Mobile Host receives any new or substantially modified route information, the new information will be retransmitted soon

(subject to constrains imposed for damping route fluctuation), effecting the most rapid possible dissemination of routing information among all the cooperating Mobile Hosts. This quick re-broadcasts introduces a new requirement for our protocols to converge as soon as possible. It would be calamitous if the movements of a Mobile Host caused a storm of broadcasting, degrading the availability of the wireless medium.

3.3 Responding to topology change

Mobile Hosts cause broken links as they move from place to place. The broken link may be detected by the layer 2- protocol, or it may instead be inferred if no broadcasts have been received for a while from a former neighbor. A broken link is described by a metric of œ (i.e., any value greater than the maximum allowed metric). When a link to a next hop has broken, any route through that next hop is immediately assigned a œ metric and assigned an updated sequence number. Since this qualifies as a substantial route change, such modified routes are immediately disclosed in broadcasting routing information packet. Sequence numbers defined by the originating Mobile Hosts are defined to be even numbers, and sequence numbers generated to indicate œ metrics are odd numbers, In this way any "real" sequence numbers will supersede and œ metric. When a node receives an œ metric, and it has a later sequence number with a finite metric, it triggers a route update broadcast to disseminate the important news about that destination.

In a very large population of Mobile Hosts, adjustments will likely be made in the time between broadcasts of the routing information packets. In order to reduce the amount of information carried in these packets, two types will be defined, One will carry will all the available routing information, called a full dump. The other type will carry only information changed since the last full dump called an incremental. By design, an incremental routing update should fit in one network protocol data unit (NPDU). The full dump will most likely require multiple NPDUS, even for relatively infrequently when no movement of Mobile Hosts occurring. When movement becomes frequent, and the size of an incremental approaches the size of a NPDE, then a full dump can be scheduled (so that the next incremental will be smaller). It is expected that mobile nodes will implement

some mean s for determining which route changes are significant enough to be sent out with each incremental advertisement. For instance, when stabilized route shows a different metric for some destination that would likely constitute a significant change that needed to be advertised after stabilization. If a new sequence number of a route is received, but the metric stays the same, that would be unlikely to be considered as a significant change.

3.4 Update a routing table

When a Mobile Hosts receives new routing information (usually in an incremental packet as just described), that information is compared to the information already available from previous routing information packets. Any route with a more recent sequence number is used. Routes with older sequence number are discarded. A route with a sequence number equal to an existing route is chose if it has better metric, and the existing route discarded, or stored as less preferable. The metric for routes chosen from the newly received broadcast information are each incremented by one hop. Newly recorded routes are scheduled for immediate advertisement tot the current Mobile Host's neighbors. Routes that show an improved metric are scheduled for advertisement at a time that depends on the average settling time for routes to the particular destination under consideration.

Timing skews between the various Mobile Hosts are expected. The broadcasts of routing information by the Mobile Hosts are to be regarded as somewhat asynchronous events, even though some regularity is expected. In such a population of independently transmitting agents, some fluctuation could develop using the above procedures for updating routes. It could turn out that a particular Mobile Hosts would receive new routing information in a pattern which causes it to consistently change routes from one next ho to another, even when the destination Mobile Host has not moved. This happens because there are two ways fro new routes to be chosen; they might have later sequence number, or they might have a better metric. A mobile Host could conceivably always received two routes to the same destination, which a newer sequence number one after another (via different neighbors), but always get the route with the worse metric first.

Unless care is taken, this will lead to a continuing burst of new route transmittals upon every new sequence number from that destination. Each new metric is propagated to every Mobile Host in the neighborhood, which propagates to their neighbors and so on.

One solution is to delay the advertisement of such routes, when Mobile most can determine that a route with a better metric. The route with the later sequence number must be available for use, but it does not have to be advertised immediately unless it is a route to a destination which was previously unreachable. Thus, there will be two routing tables kept at each Mobile Host; one for use with forwarding packets, and another to be advertised via incremental routing information showing a better metric, the Mobile Hosts has to keep a history of the weighted average time that route to a particular destination fluctuate until the route with the best metric is received.

3.5 Damping Fluctuations

The following describes how the settling time table is used to prevent fluctuations of routing table entry advertisements. The general problem arises because route updates are selected according to the following criteria:

- Routes are always preferred if the sequence numbers are newer.
- Otherwise, routes are preferred if the sequence numbers are the same and yet the metric is better (lower).

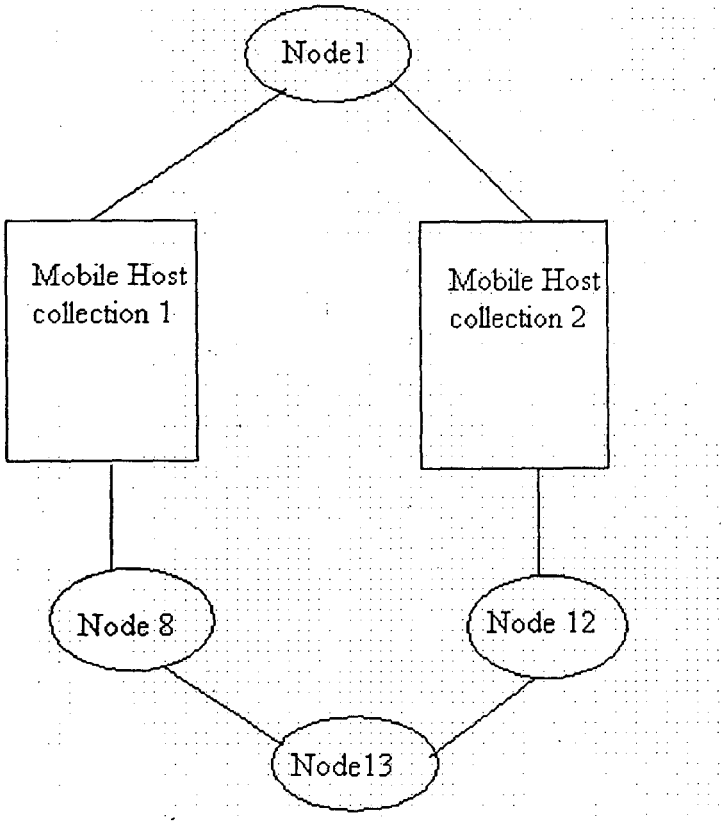


Figure3.1: Receiving fluctuating routes

To see the problem, suppose the two routes with identical sequence numbers are received by a Mobile Host, but in the wrong order. In other words, suppose that node15 receives the highest metric next hop first and soon after gets another next hop with a lower metric but the same sequence number. This could happen when there are a lot of Mobile Hosts, transmitting their updates not quite regularly. Alternatively, if the Mobile Hosts are acting independently and with markedly different transmission intervals, the situation could occur with correspondingly fewer hosts. Suppose, in any event, in Figure that there are enough Mobile Hosts to cause the problem, in two separate collections of Mobile Hosts both connected to a common destination Nodes1, but with no other Mobile Hosts in common. Suppose further that all Mobile Hosts are transmitting updates approximately every 15 seconds, that Mobile Host nodes has a route Node1 with 12 hops, and Mobile Host Node12 has a route to Node1 with 11 hops, Moreover, suppose that the routing

information updates from Node8 arrives at Node15 approximately 10 seconds before the routing information update from Node12. This will occur every time that a new sequence number is issued from Mobile Host Node1. In fact, the time differential can be drastic if any Mobile Host in collection 2 begins to issue its sequence number updates in multiple incremental update intervals, as would happen, for instance, when there are too many hosts with new sequence number updates for them all to fit within a single incremental packet update. In general, the larger the number of hops, the more drastic differentials between deliveries of the updates can be expected in Figure.

The settling time data stored in a table with the following fields, keyed by the first field:

- Destination address
- Last settling time
- Average settling time

The settling time is calculated by maintaining a running, weighted average over the most recent updates of the ruts, for each destination.

Suppose a new routing information update arrives at Node15. The sequence number in the new entry is the same as the sequence number in the currently used entry, and the newer entry has worse (i.e., higher) metric. The Node15 must use the new entry in making subsequent forwarding decisions. However, Node15 does not have to advertise the new route immediately and can consult its route settling time table to decide how long to wait before advertising it. The average settling time is used for this determination. For instance, Node15 may decide to delay (average_setting_time X2) before advertising a route.

This can be quite beneficial, because if the possibly unstable route were advertised immediately, the effects would ripple through the network, and this bad effect would probably be repeated every time Mobile Host Node 1's sequence number updates rippled through the ad-hoc network. One the other hand, if a link via Mobile Host Node12 truly

does break, the advertisement of a route via Node8 should proceed immediately. To achieve this when there is a history of fluctuations at Mobile Host Node15, the link breakage should be detected fast enough so that an intermediate host in collection 2 finds out the problem and begins a triggered incremental update showing an œ metric for the path along the way to Mobile Host Node1. Routes with an metric are required by this protocol to be advertised immediately, without delay.

In order to bias the damping mechanism in favor of recent events, the most recent measurement of the settling time of particular route must be counted with a higher weighting factor that is less recent measurements. And, importantly, a parameter must be selected which indicates how long a route has to remain stable before it is counted as truly stable. This amounts to specifying a maximum value for the settling time for the destination in the settling time table. Any route more stable that this maximum value will cause a triggered update if it is ever replaced by another route with a different next hop or metric.

When a new routing update is received from a neighbor, during the same time that the updates are applied to the table, processing also occurs to delete stale entries. Stale entries are defined to be those for which no update has been applied within the last few update periods. Each neighbor is expected to send regular up-dates; when no updates are received for a while, the receiver may make the determination that the corresponding computer is no longer a neighbor. When that occurs, any route using that computer as a next hop should be deleted, including the route indicating that computer as the actual (formerly neighboring) destination. Increasing the number of update periods that may transpire before entries are determined would result in more stale routing entries, but would also allow for more transmission errors. Transmissions errors are likely to occur when a CSMA-type broadcasts medium is used, as may well be the case for many wireless implementations. When the link breaks, asymmetric route should be advertised for it, as well as for the routes that depends on it.

There are additional data fields, other than those stated above, which might be transmitted as part of each entry in the routing tables which are broadcast by each participating computer (mobile or base station). These fields may depend, for instance, on higher-level protocols or other protocols depending on the operation of layer 2. For instance, to enable correct ARP operation, each routing table entry must also contain an association between the Internet Protocol (IP) address and the MAC address of the destination. This would also enable an intermediate computer, when serving a routing function for its neighbors, to also issue proxy ARP replies instead of routing ARP broadcasts around. However, if packet forwarding is based on MAC addresses, hopefully such techniques will be unnecessary. And, if forwarding is done based on IP addresses, no ARP is strictly necessary, as long as neighboring nodes keep track of associations gleaned from route table broadcasts. Note also that layer 3 operation violates the normal subnet model of operation, since even if two Mobile Hosts share the same subnet address there is no guarantee they will be directly connected- in other words, within range of each other.

3.6 Properties of the DSDV Protocol

Potentially a loop may form each time a node i change its next hop. This can happen in two cases:

- Node i detect that the link to its next-hop is broken clearly, this action cannot form a loop involving i.
- Node i receives from one its neighbor's K, route to D, with a sequence numbers

  (k) Such that s (k)>s (i) (where s(i) is sequence number for the Destination D as originally stored in node i).

A node i prorogates sequence number s(i) to its neighbors only after receiving it from its current next hop. When a node i choose a particular neighbor as the next ho for a particular destination, the information for this is received by that next hop only. So, at all times the sequence number value stored at the next ho is always greater or equal to the value stored at i. starting from node i, if we follow the chain of next-hop pointers, the sequence number values stored at visited nodes would form a non decreasing sequence. Now suppose, node i form a loop by choosing K as its next ho. This would imply node i life before and after K. Since it lies after K, thus we must have s(k) <=s(i) (as proved that sequence numbers along a path form no decreasing sequence). But this contradicts the initial assumption that s(k)>s(i). Hence, loop formation cannot occur if nodes use never sequence numbers to pick routes.

CHAPTER-4

SIMULATION ENVIRONMENT AND METHODOLOGY

The network simulator should provide the mobility features to simulate the networks like ad hoc networks, which has dynamic topology in which nodes may enter or leave the network at any time. The network Simulator, NS2 is the one that is the result of an effort of research and development that is administrated by researches at UC, Berkeley.

4.1 Network Simulator

NS is a discrete event simulator targeted at networking research. NS simulates variety of IP networks. It implements network protocols such as TCP and UDP, traffic source behavior such as FTP, Telnet, Web, CBR, and VBR, router queue management mechanism such as Drop Tail, RED and CBQ, routing algorithm such as Dijkstra, and more. NS also implements multicasting and some of the MAC layer protocols for LAN simulations. The simulator is written in C++ and a script language called OTcl [7].

NS is an Object Oriented Tcl (Otcl) script interpreter that has an event scheduler, network component objects and network setup modules called plumbing modes. The user writes and Otcl script that defines the network (number of nodes, links), the traffic in the network (source, destinations, type of traffic) and which protocol sit will use. The OTcl script initiates an event scheduler, sets up the network topology using the network objects and the plumbing functions in the library, and tells traffic sources when to start and stop transmitting packets through the even scheduler. One can create new object either by writing a new object or by making a compound object form the object library, and plumb the data path through the object [4].

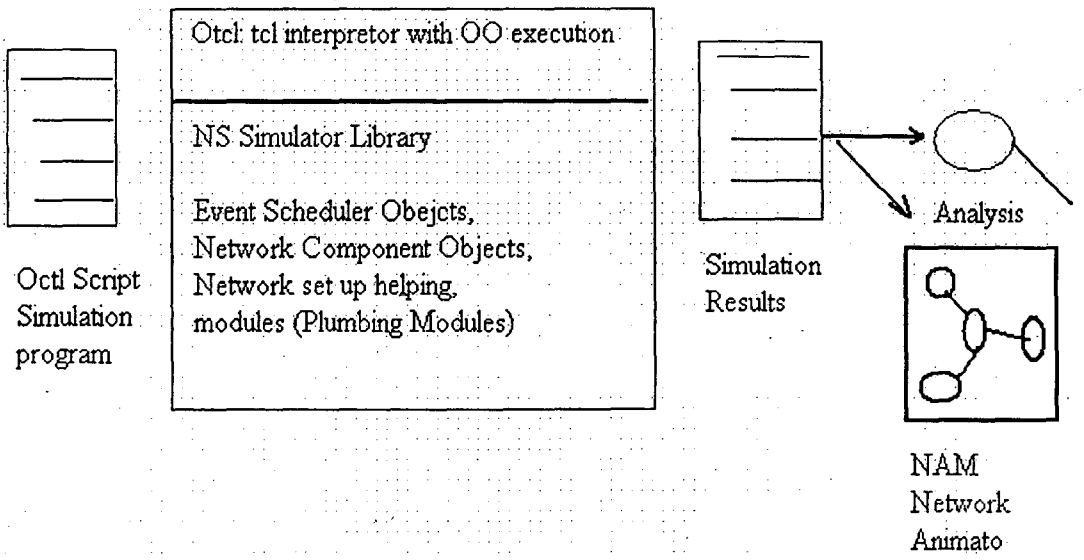


Figure 4.1: Simplified User view of NS

An event is NS is a packet ID that unique for a packet with scheduled time and the pointer to an object that handles the event. In NS the event scheduler keeps track of simulation time and files all the vents in the event queue scheduled for the current time by invoking appropriate network components, usually the ones who issued the vents, and left they do the appropriate action associated with packet pointed by the event. Network components communicate with one another passing packets, however this does not consume actual simulation time. All the network components that need to spend some simulation time handling the packet use the event scheduler by issuing an event for the packet and waiting for the event to be fired to itself before doing further action handling the packet. For example, a network switch component that simulates a switch with 15 microseconds of switching delay issues an event for packet to be switched to the

scheduler as an event 15 microsecond later. The scheduler after 15 microsecond deques the event and fires it to the switch component, which the passes the packet to an appropriate output link component. Another use of event scheduler is timer. For example, TCP needs a timer to keep track of a packet transmission time out for retransmission. Timers measures a time value associated with a packet and does an appropriate action related to that packet after a certain time goes by, and does not simulate a delay.

NS separates the data path implementation from control path implementation. In order to reduce packet and event processing time (not simulation time), the vent scheduler and basic network component objects in the data path are written and compiled using C++. The compiled objects are made available to the OTcl interpreter thorough an OTcl linkage that creates a matching OTcl object for each of the C++ objects. The Otcl linkage also makes the control functions and the configurable variables specified by the C++ objects as a member functions variables of the corresponding OTcl object.

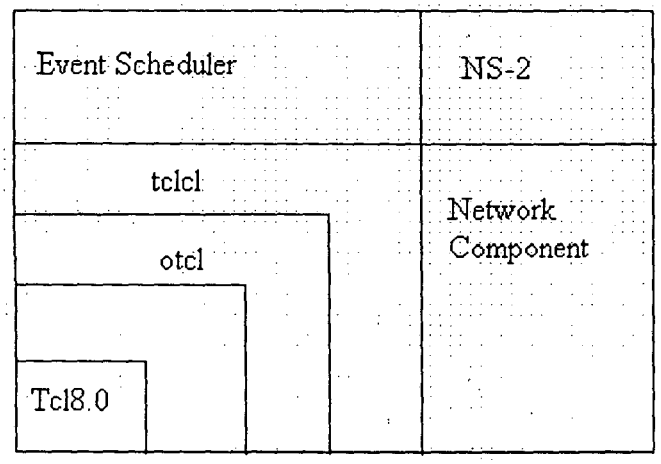


Figure 4.2: Architectural View of NS

The above figure shows the general architecture of NS. A general user can be thought of standing at the left bottom corner, designing and running simulations in Tcl using the simulator objects in the OTcl library. The event schedulers and most of the network components are implemented in C++ and available to OTcl through and OTcl linkage that is implemented using tclcl.

When a simulation is completed, NS produces one or more text-based output files called trace files that contain the detailed simulation date. The data can be used for simulation analysis. The out put data can be used as input to the graphical simulation display tool called Network Animator (NAM). Nam has a nice graphical user interface similar to that of a CD player which provides play, fast-forward, rewind, pause button options. The network animator also graphically presents the information such as throughput and number of packet drops at each link, although the graphical information cannot be used for accurate simulation analysis.

4.2 Mobility Node

Mobile Node is the basic NS node object with added functionalities like ability to transmit and receive on a channel that allows it to be used to create mobile, wireless simulation environments. The mobility features including node movement, periodic position updates, maintaining topology boundary etc are implemented in C++ while plumbing of network components within Mobile Node itself have been implemented in Otcl [12].

Each node is mobile and independent entity that is responsible for computing its own position and velocity as a function of time. The mobility features including node movement, periodic updates, maintaining topology boundary etc are implemented in C++ while plumbing of network components within Mobile Node itself have been implemented in OTcl [7].

Each Mobile Node uses a routing agent for the purpose of calculating routes to other nodes in the ad hoc networks. Packets are sent from the application and are received by the routing agent. The agent decides a path that the packet must travel in order to reach its destination and stamps it with this information. It then sends the packet down to the link layer. The link layer level uses an Address Resolution Protocol (ARP) to decide the hardware addresses of neighboring nodes and map IP address to their correct interfaces. With this information the packet goes to the interface queue and awaits a signal from the Multiple Access Control (MAC, IEEE 802.11) protocol. When MAC layer decides it is ok to send it onto the channel, it fetches the packet from the queue and hands it over to the network interface that in turn sends the packet onto the radio channel. Then the packet is delivered to all network interfaces. Each network interface stamps the packet with the receiving interfaces properties and then invokes the propagation modes.

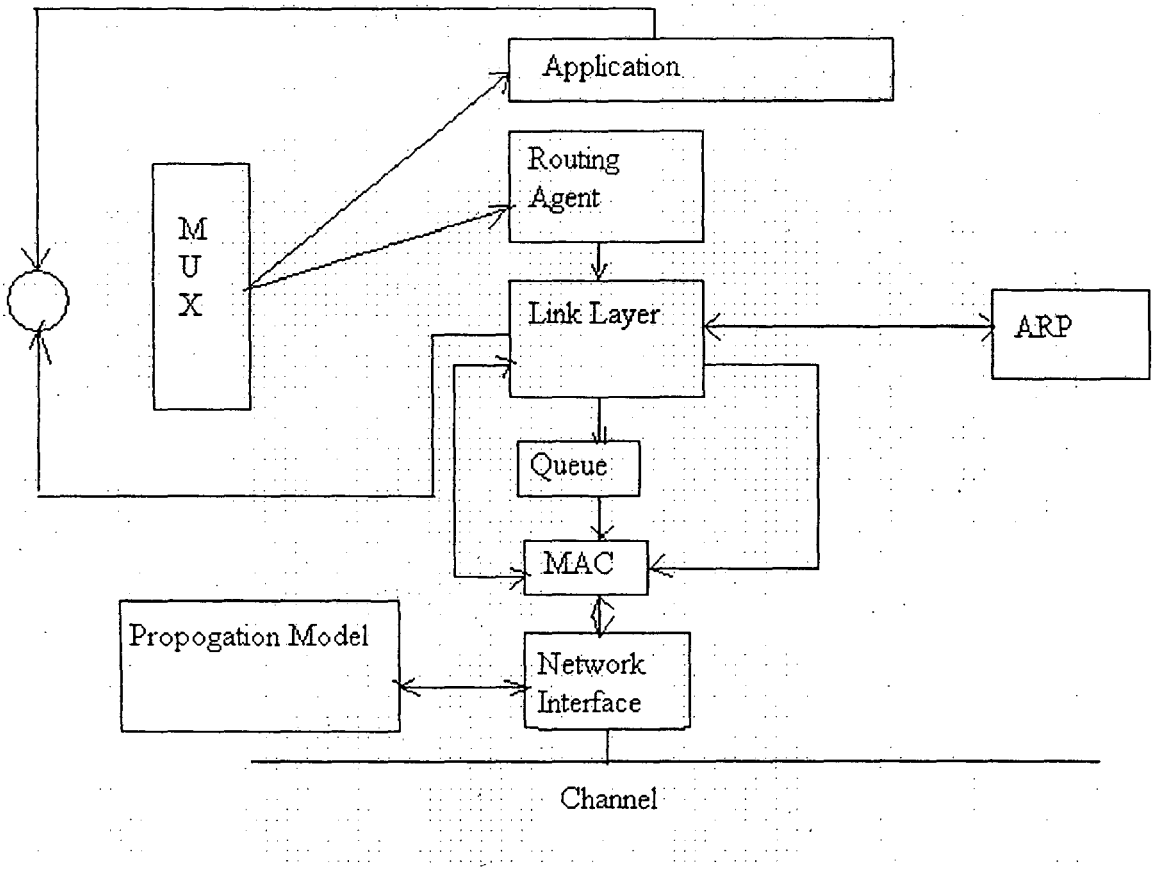


Figure4.3: A mobile node

4.3 Methodology

In this study an ad hoc network that networks on dynamic source routing and destination sequenced distance vector algorithms have been simulated using the ns2 .For creating a mobile node NS2 provides with the class Mobile node .The characteristics of the mobile node can be set using the config( ) function defined on the mobile node .The parameters like routing protocol being used . The link layer protocol on which the algorithm works, battery power used for transmission of data and for receiving of data etc, can be set using this class. Whenever a node is created in NS using this mobile node class, all the parameters are set on the node. There after the nodes behave as per their specifications [11].

As ad hoc networks are highly mobile in nature, there should be a support for the high mobility of the nodes .The movement of the nodes can be configured in such a way that the network is highly mobile and ad hoc in nature. NS is designed in such way that the nodes can enter and leave the network at any time .For giving the dynamic nature to the algorithm ns makes use of two files called mobility file and traffic pattern file. The Network simulator uses the mobility file to induce the node movement in to the network and the traffic pattern file is used to introduce the traffic across the various wireless nodes.

The rate with the nodes move can be specified in the mobility file and the traffic pattern can be specified in the traffic pattern file like burst traffic or normal traffic by specifying the number of connections across the network nodes. Like wise the battery power, the initial energy model .The power that was consummated when packet is received by the mobile node can be fixed in the NS2 by using the energy model. The power that was consummated when transferring data packet out of the mobile node and the power that was consummated when packet is received by the mobile node can also be fixed using the energy model.

The methodology followed in the process of running the simulator that is explained before in this chapter. In the process of running the simulator, first a simulation script is written which is input to the NS and initiates the process of simulation. The script is written for creating creating the nodes, establishing the links, specifying the traffic patterns, specifying the node movement scenarios etc., energy model also induced in to the network, which handles the battery power. The power with which the battery can transmit and receive the signals can be varied using the EnergyModel class defined in NS2.

############## TO SET THE PARAMETERS FOR THE NETWORK ############

```
set val(chan)   Channel/WirelessChannel
set val(prop)   Propagation/TwoRayGround
set val(ant)    Antenna/OmniAntenna
set val(ll)      LL
set val(ifq)    Queue/DropTail/PriQueue
set val(ifqlen)  50
set val(netif) Phy/wirelessPhy
set val(mac)  Mac/802_11
set val(rp)    DSR/DSDV
set val(nn)    50
set val(x)     670
set val(y)     670
set val(stop)  200.0
set val(tr)    /root/anil/test/anilkr.tr
set val(nam)  /root/anil/test/anilkr.nam
set val(traffic)  "/root/ns2/ns-allinone-2.26/ns-2.26/tcl/mobility/scene/test-cbr-50-5"
set val(movement) "/root/ns2/ns-allinone-2.26/ns-2.26/tcl/mobility/scene/mv-50-5"
```

The command set sets the values to the parameters. For example set val(ifqlen) 50, sets the value of the queue length to 50. In an ad hoc network, every node maintains a queue to handle packets. As the queue length is set to 50, during the simulation the queue can handle a maximum of 50 packets only. If the queue is full, then the node starts dropping packets.

The simulation is run for a number of nodes varying from 25 to 100 with an increment of 25 nodes, for a traffic pattern of 5bps (normal traffic),50bps and 120bps(bursty traffic), and for a mobility rate of 1m/s, 5m/s, 10m/s, 25m/s. When the network size is varied, the traffic pattern and mobility rate are fixed at 5bps and 5m/s respectively. When the traffic pattern is varied, the network size is fixed at 50 and the mobility rate at 5m/s. When the mobility rate is varied, the network size is fixed at 50 and the traffic pattern is fixed at 5bps.

The other parameters that are fixed during the entire simulation are,

Channel----------Wireless Channel
Propagation Model----------TwoRayGround
Antenna ----------OmniAntenna
Queue----------DropTail/PriQueue
QueueLength----------50
PhysicalLayer----------WirelessPhysical
Mac----------802.11

Once the algorithms have been simulated and the results of the simulation are obtained, then the performance of the algorithms has to be evaluated on the parameters defined below:

Throughput: This is defined as the ratio of numbers of data packets received by the destination to the number of data packets sent by the source per unit time.

Overhead ratio: It can be defined as the ratio of total control bits transmitted to actual data bits transmitted.

Power consumption: It measures the average power consumption per node.

The simulations are run in a grid size of 670X670 and for a time period of 200 seconds with the fixed bandwidth of 1 Mbps. The initial energy of battery is set at 20 J, the transmission power at 5milliwatts and the receiving power at 1 milliwatt.

The mobility and the traffic pattern are created using the facility provided by NS. The random traffic connections of TCP type or CBR type can be created using the cbrgen.tcl file, which comes with NS2, and the node movement scenarios can be generated using *setdest* command in NS. In the present simulation CBR (constant bit rate traffic) is used.

The results of the simulation are two files which contains the resulted data. The output files are called as *trace* file and *nam* file. The *nam* file can be give as an input to the network animator, which comes as a part of NS2, to view the simulation graphically. The *trace* file can be used to analyze the simulation results. Each time a simulation is run then the results are written to the trace file specified.

After running the Network Simulator the trace files are studied properly and routines are written to extract the required data from those files. Based on that extracted data for different parameters, the graphs are drawn.

CHAPTER-5

SIMULATION STUDY

The previous chapter described the methodology for performing the simulation and discussed the basic steps involved in the process of simulating the network, under the network simulator environment. The current chapter discusses how the simulation is done and the simulation results are analyzed.

5.1 Simulation overview

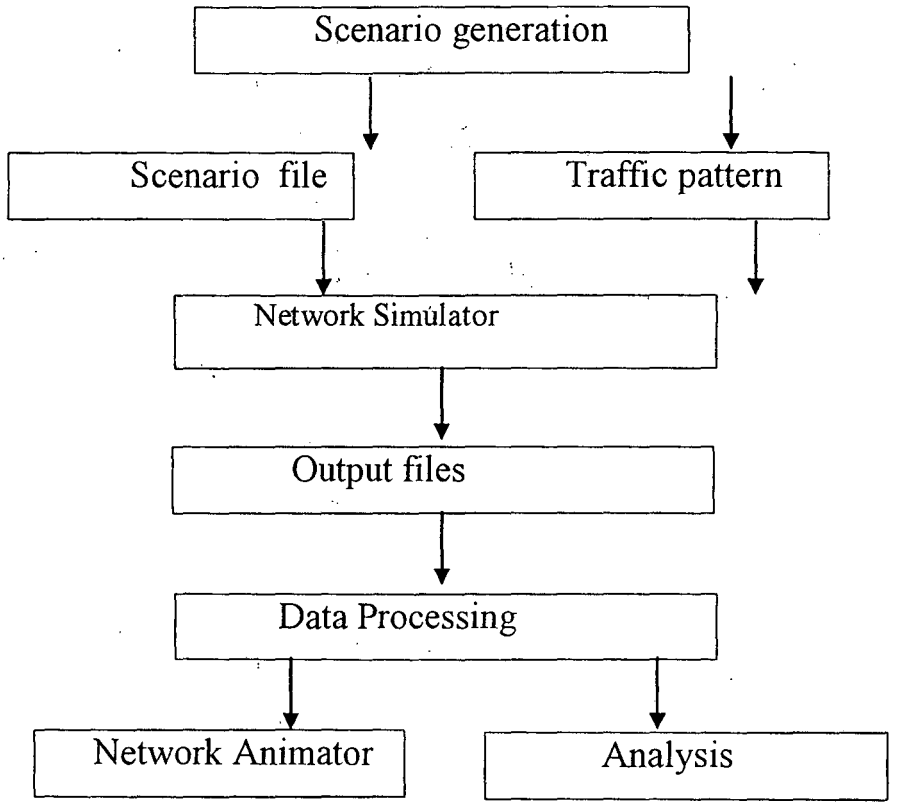


Figure: Simulation overview.

The above figure shows the overview of simulation process done in this work. First the scenario files will be generated which are used for node mobility and traffic pattern as mentioned above. The OTcl script with the modified node configuration uses these

scenario files. Then the OTcl script will be given as an input to the network simulator. The output files that result from the simulation process will be given as input to the Network Animator. By using the data from the output file, the above mentioned parameters will be evaluated.

5.2 Simulation results

5.2.1 Throughput

It is referred as the ratio of number of data packets received by the destination to the number of data packets sent by the source per unit time. Throughput measures the effectiveness of the network in delivering data packets. That is, how well does the network deliver packets from source to destination?

Based on the values calculated for throughput for a node , a graph is drawn between the network size (number of nodes in the network) on the X axis and the ratio of received data packets by the destination node to the sent data packets by the source node (i.e. throughput) on the Y axis.

For a varying network size of 25 nodes, 50 nodes, 75 nodes, 100 nodes, the graph drawn is shown below:

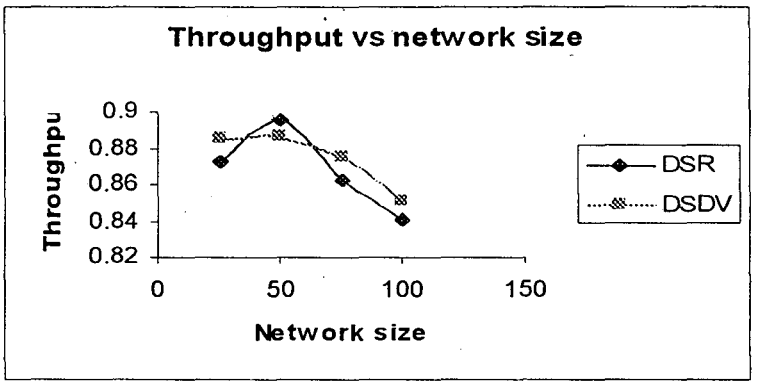


Figure5.1: Throughput vs. network size

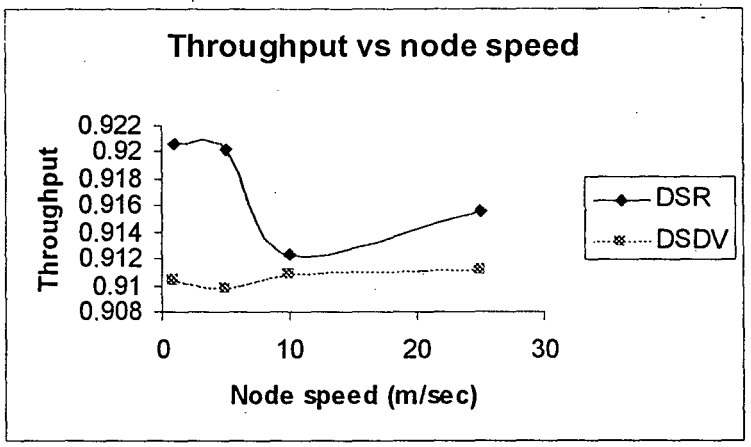


Figure5.2: Throughput vs. node speed

In the graph shown in Figure5.1 for DSR, when the network size varies from 25 nodes to 100 nodes, the delivery rate is varying. The network shows no significant trend. The value is very small for 100 nodes network, this may be due to congestion in the network. In the case of 25 nodes network, the nodes may be scattered and thus the rate decreases. The increment in the values of network of 50 nodes and 75 nodes may be due to the availability of alternative route as the network is growing. Even though some nodes are dropping packets, the available routes may pass them to the destination which may lead to an increase in the value. As the number of mobile nodes in the network is increased, the delivery rate of packet decreases in the DSDV protocol

Similarly, based on the values calculated for throughput for a node a graph is drawn between the node speed on the X axis and the throughput on the Y axis, for a varying mobility rate of 1m/sec, 5m/sec, 20m/sec, and 25m/sec. The graph drawn is shown here in Figure5.2. In DSR, a sharp decline in the throughput can be seen for the nodes moving with a rate 5 m/sec to 10 m/sec. if the nodes move with a rate of 25 m/sec then there is a possibility for the network partition, which will lead to the decrease in the results.

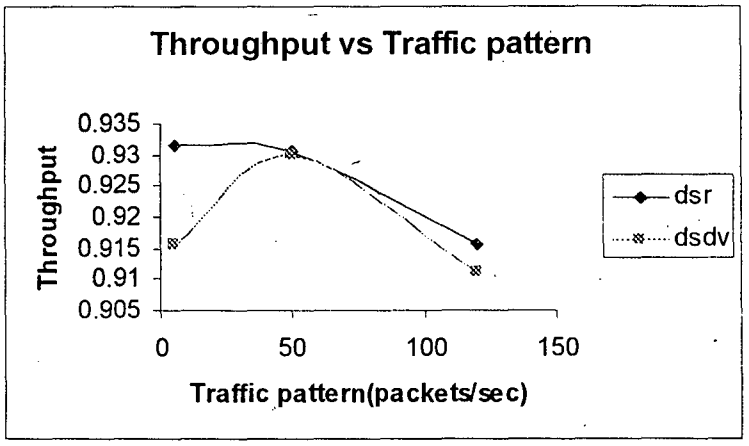


Figure5.3: Throughput vs. traffic pattern

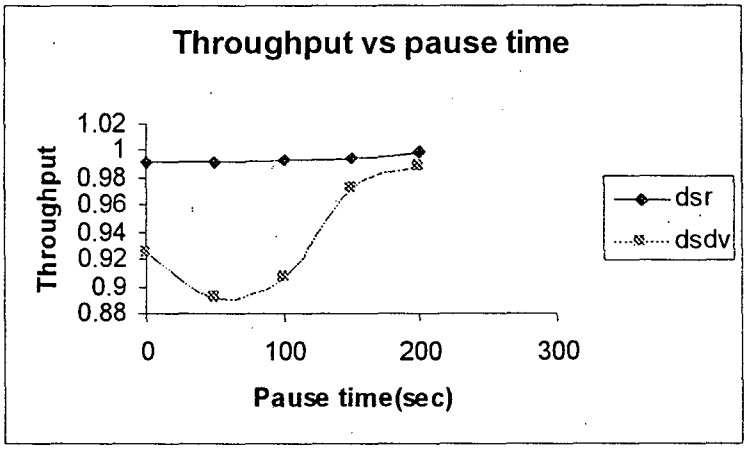


Figure5.4: Throughput vs. pause time

In DSDV, increased mobility rate causes decrease in successful packet delivery especially for protocols, which demand on control messaging for link breakages. This happens because of decreasing reliability of such control messages transmitted in IP level, and mainly due to their losses. DSDV is a bidirectional protocol that maintains their successful delivery rate due to their simple management of route maintenance. Proactive protocol suffers for their lateness in adapting to changing topology. Next when the traffic pattern increases in the network then the control packets also increases in the network resulting throughput decreases due to congestion in the network as shown in the Figure5.3.

At a pause time of 0 all the nodes in the network are in constant motion. As the pause time increases from left to right, the average node movement rate in the network decreases. At a pause time of 200, all nodes are stationary because each simulation is run for 200 simulated seconds of operation of the ad hoc network. So we can say that when the pause time increases, the throughput for both of the protocol increases as shown in Figure 5.4.

5.2.2 Overhead ratio

The routing overhead is the number of routing overhead packets generated by DSR and DSDV.

When the size of network increases the number of control packets to update the tables, results increasing in overhead as shown in the Figure5.5. When the speed of nodes increases the performance of DSR and DSDV in terms of routing overhead increase as shown in Figure5.6.

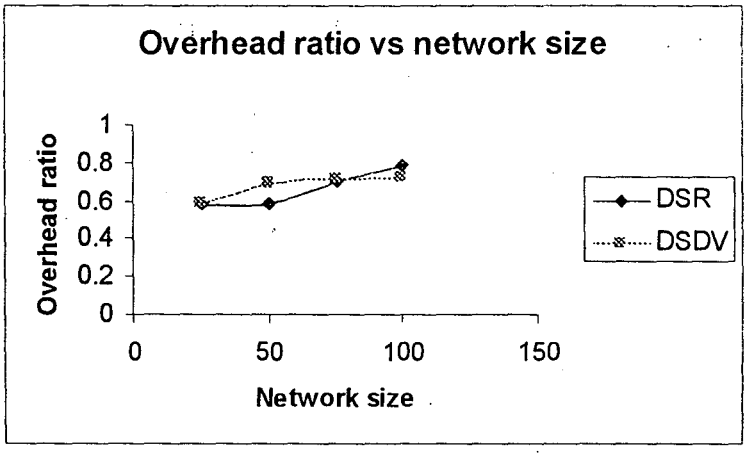


Figure5.5: Overhead ratio vs. network size

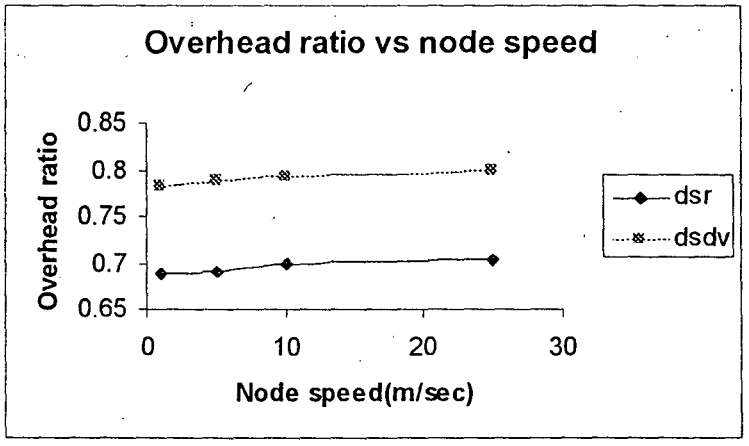


Figure5.6: Overhead ratio vs node speed

Similarly when number of sources with varying traffic pattern increases in the network then the control packets also increase resulting the routing overhead also increase for DSR but not always as shown in Figure5.7. When the traffic is normal then the overhead is normal but when the traffic pattern increases then the overhead ratio also increases for a limited value of traffic pattern. And when the traffic pattern is bursty then the overhead ratio decreases. In DSDV the overhead ratio also decreases when the traffic pattern increases in the network. Similarly for the pause time, when pause time increases then the control packets decreases resulting the overhead ratio decreases for both of the routing protocol as in Figure5.8.

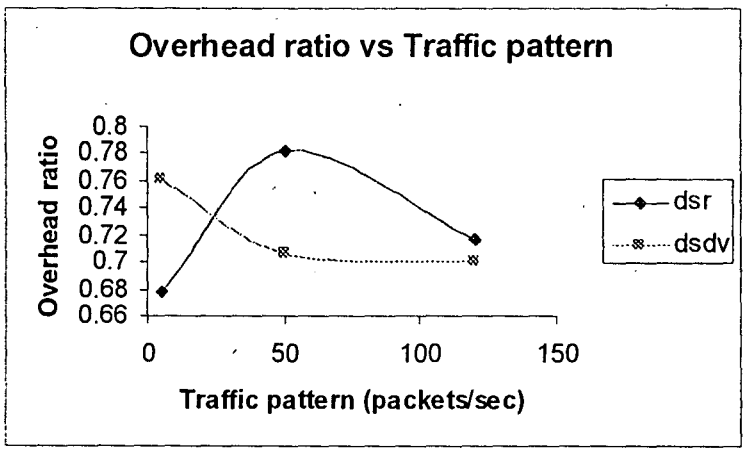


Figure5.7: Overhead ratio vs. traffic pattern

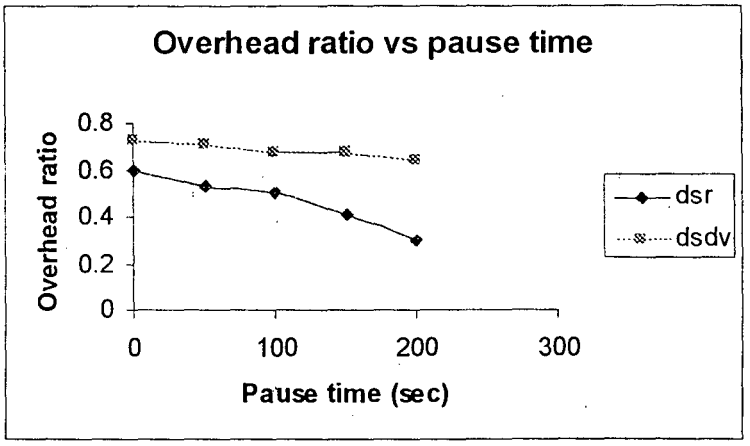


Figure5.8: Overhead ratio vs pause time

5.2.3 Power consumption

This measures the average power consumption per node as energy is a limited resource in the ad hoc networks. Power consumption is strongly influenced by two factors. First factor is network size, when then network size increases then the power consumption decreases in both of the routing protocol. The Figure5.9 -5.12 highlights the average energy consumed to transmit a CBR data stream due to routing packets. The results show that DSR offers the best performance while the DSDV protocol shows an energy consumption practically constant as motion rate varies-this is because of their table – driven philosophy.

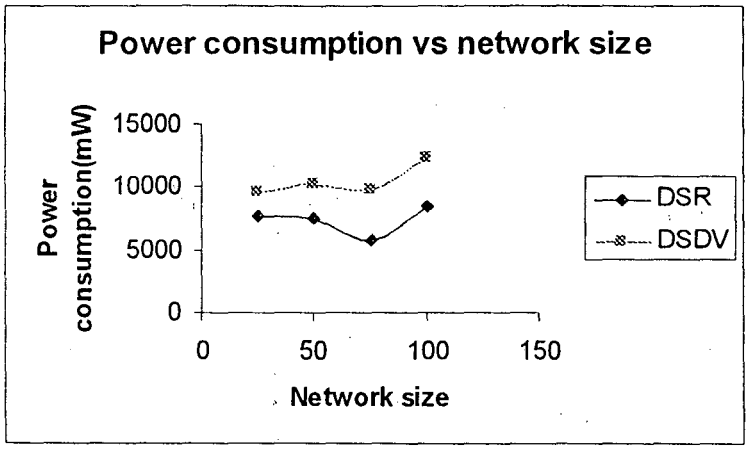


Figure5.9: Power consumption vs. network size

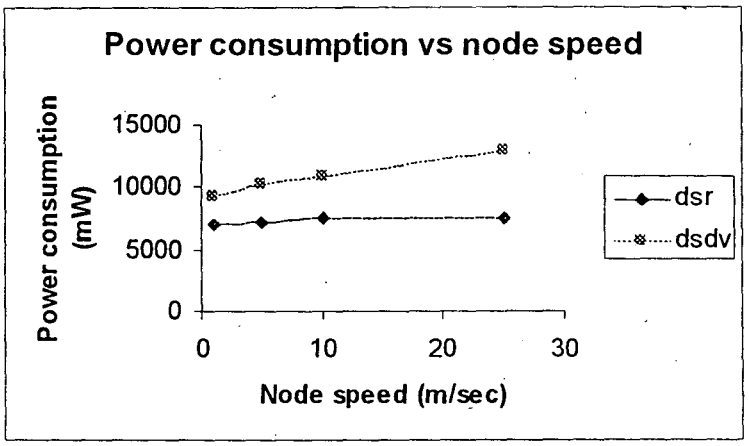


Figure5.10 Power consumption vs node speed

Second factor is traffic pattern (no. of traffic sources), as the number of traffic pattern increases the average power consumption increases. This is reasonable since increasing the network size while maintaining a constant traffic load essentially increases the number of routers eligible to forward packets, resulting in a reduced load.

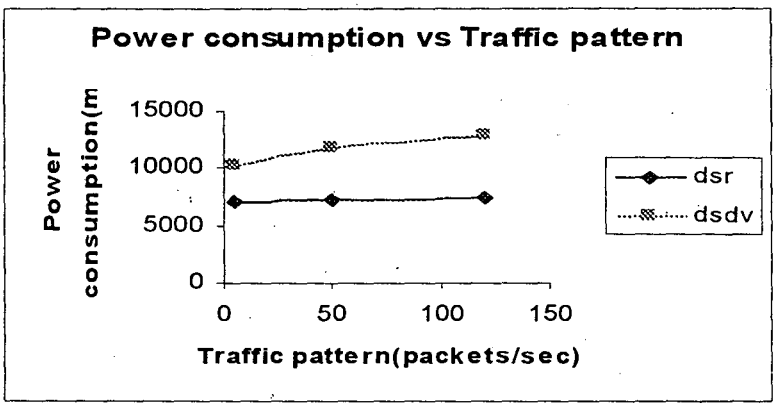


Figure5.11: Power consumption vs. Traffic pattern

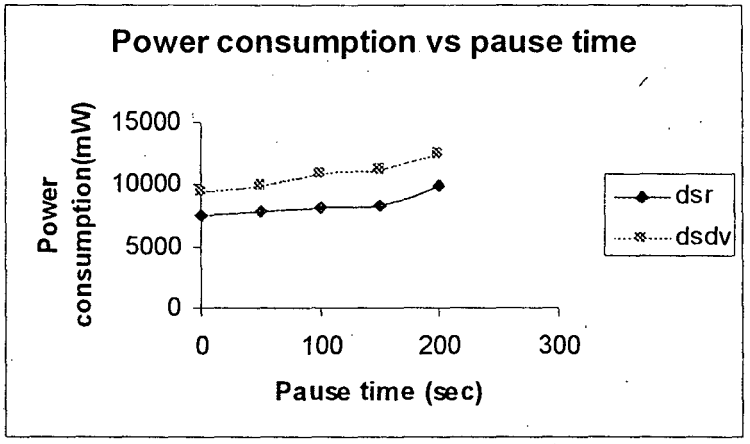


Figure5.12: Power consumption vs pause time

On the other hand, increasing the number of traffic sources simply increases the routing load of each mobile host, resulting in increased power consumption. By analyzing the simulation the main effect of node speeds and pause time have no real impact on power consumption. Figure shows again the total energy consumption per packet correctly transmitted as a function of pause time, the pure on demand protocol DSR perfórms better than DSDV

CONCLUSION AND FUTURE WORK

In the present work simulation has been done to analyze the performance of DSR and DSDV in terms of average throughput, routing overhead ratio and power consumption. The results of simulation are presented in the form of graphs so that the conclusions can be made by analyzing the trends shown by these graphs. But it has been difficult in some cases to make definite conclusion. This may require more efforts to consider other network context so that better conclusion can be made.

Analyzing the graphs it can be concluded that increase in speed, network size, number of resources, and the average throughput in the ad-hoc networking decreases.
Similarly increasing in the speed ,network size, number of resources, the routing overhead increases for both of the protocols but in some cases when the of resources is increased then the routing overhead decreases for DSDV. Similarly for power consumption, we presented the results of measuring and comparing the power consumption behavior in a mobile ad hoc network using two routing protocols DSR and DSDV. Focusing on routing protocol details, simulation shows that as far as power consumption refers, DSR performs better than DSDV. DSDV is quite stable regardless of the used traffic load whereas DSR produces an increment of routing packets by increasing the number of sources.

The limitations of the present work can be overcome by extending it in future. The performance of algorithm can be analyzed for the other parameters that have not been considered in this present work.

BIBLIOGRAPHY

[1] Andrew S. Tanenbam, "Computer Networks", PHI publications, 3rd edition 2000.

[2] Charles E.Perkins, Elizabeth M. Royer , Samir R.Das ," Ad Hoc on demand Distance Vector (AODV) Routing " , Internet Draft , March 2001 , www.left..org/internet-drafts/draft-ietf-manet-manet -aodv-08.txt.

[3] Charles E.Perkins, P.Bhagwat," Highly dynamic DSDV for Mobile Computer ", SIGCOM '94 conference on Communication Architecture, protocols and Applications www.cs.umd.edu/projects/meml/papers/Sigcomm94.ps.

[4] Charles E. Perkins, "Mobility support, Mobile IP and Wireless Channel Support for ns-2" RFC 2002(proposed standard) IETF ,Oct 1996, www.svrloc.org/~charliep/mobins2

[5] Charles E. Perkins, "Mobile Ad hoc Networking Terminology" , Internet Draft from IETF MANET Working Group, Oct 1997 www.ietf.org/internet-drafts/draft-ietf-manet-term-01.txt

[6] David B Johnson, Dravid A. Maltz , Yih-Chun, Jorjeta G. Jetcheva " The Dynamic Source Routing algorithm for mobile Ad hoc networks " , Internet Draft from IEFT MANET working Group ,Oct 1999 www.ietf.org/internet-drafts/draft-ietf-manet-dst-05.txt.

[7] Jae Chung Mark Claypool "NS by Example ", WPI Computer Science, www.nile.wpi.edu/NS.

[8] J. Broch, D.Maltz, D. Johnson, Y.HU, J. jetcheua,"A Performance Comparision of Multi – hop Ad-hoc Network Routing Protocols ", ACM/IEEE international Conference on Mobile Computing and Networking' Oct 1998.

[9] Jochen H Schiller, "Mobile Communication", PHI publications

[10] J. Maker, S. Corson, "Mobile Ad Hoc Networks (MANET): Routing protocol Performance Issues and Evaluation considerations", RFC 2501, www.ietf.org/rfc/rfc2501.txt

[11] Kevin Fall, Kannan Varadhan," Ns Manual ", VINT project, May 2001, www.manch.cs.berkeley.edu/ns/ns-man.html.

[12] Mark Gries "Tutorial for Network Simulator NS-2 ", www.isi.edu/nsnam/ns/tutorial.

[13] Nitin H. Vaidya. Texas A&M University "Mobile Ad hoc Networks: Routing, Mac and Transport issues "www. cs. Tamu.edu/faculty/vaidya.

[14] Santhosh R. Thampuran," Routing Protocols for Ad hoc Networks of Mobile Nodes ", www. UNIX .ecs.umass.edu/-sthampur/Papers/Ad hocRouting.pdf (Mobicom 98).

Tabular data of Simulation Results

The following are the results obtained at the end of simulations:

1. The following are the simulation results for DSR when network size is increasing then the results looks like that:

Observed parameters	For a network of 25 nodes	For a network of 50 nodes	For a network of 75 nodes	For a network of 100 nodes
No. of pkt received (r)	1621	1272	1422	1783
No. of pkt sent (s)	1858	1420	1650	2122
Throughput (r/s)	0.87244	0.89577	0.86181	0.84024
DSR control pkt (c)	4892	4060	6092	5112
Ack pkt (a)	8636	7224	7437	7620
Data pkt (d)	23074	19116	19403	16213
Overhead ratio	0.586271	0.59029	0.69726	0.78529
Power consumption	7669	7498	5810	8490

2. When speed of nodes is increasing then the results looks like that:

Observed parameters	For a mobility rate of 1 m/sec.	For a mobility rate of 5 m/sec.	For a mobility rate of 10 m/sec.	For a mobility rate of 25 m/sec.
No. of pkt received (r)	1566	1672	1613	1694
No. of pkt sent (s)	1701	1817	1768	1850
Throughput (r/s)	0.92063	0.92019	0.91233	0.91567
DSR control pkt (c)	2209	1039	2625	5331
Ack pkt (a)	9450	9326	8054	8667
Data pkt (d)	16926	15025	15271	19936
Overhead ratio	0.68882	0.68985	0.69929	0.70214
Power consumption	7090	7215	7498	7556

3. When traffic pattern is changing then the results looks like that:

Observed parameters	For a traffic pattern of 5 packets/sec.	For a traffic pattern of 50 packets/sec	For a traffic pattern of 120 packets/sec
No. of pkt received (r)	1714	2094	2234
No. of pkt sent (s)	1840	2250	2440
Throughput (r/s)	0.93152	0.93066	0.91557
DSR control pkt (c)	6027	8927	7886
Ack pkt (a)	9325	7239	4941
Data pkt (d)	22646	20651	17894
Overhead ratio	0.67791	0.78281	0.71683
Power consumption	6972	7210	7396

4. When pause time is increasing then the results becomes like that:

Observed parameters	When pause time 0 sec.	When pause time 50 sec.	When pause time 100 sec	When pause time 150 sec.	When pause time 200 sec.
No. of pkt received (r)	1935	1752	2070	1848	1767
No. of pkt sent (s)	1950	1766	2084	1857	1768
Throughput (r/s)	0.99230	0.99207	0.99328	0.99515	0.99943
DSR control pkt (c)	2152	1869	2358	2056	2218
Ack pkt (a)	8367	8513	7926	8398	8224
Data pkt (d)	17571	19657	20524	25464	34641
Overhead ratio	0.59865	0.52815	0.50107	0.41054	0.30143
Power consumption	7510	7880	8079	8220	9866

5. The following are the simulation results for DSDV. When network size increasing then the results looks like that:

Observed parameters	For a network of 25 nodes	For a network of 50 nodes	For a network of 75 nodes	For a network of 100 nodes
No. of pkt received (r)	1936	2175	2022	2551
No. of pkt sent (s)	2187	2453	2310	2998
Throughput (r/s)	0.88523	0.88666	0.87532	0.85090
DSR control pkt (c)	8512	9320	8917	9217
Ack pkt (a)	7264	8113	8019	7983
Data pkt (d)	26871	25163	23690	23952
Overhead ratio	0.5871	0.6928	0.7149	0.7181
Power consumption	9566	10183	9685	12266

6. When the speed of nodes is increasing then the results looks like that:

Observed parameters	For a mobility rate of 1 m/sec.	For a mobility rate of 5 m/sec.	For a mobility rate of 10 m/sec.	For a mobility rate of 25 m/sec.
No. of pkt received (r)	2012	2229	2288	2746
No. of pkt sent (s)	2210	2450	2512	3014
Throughput (r/s)	0.91040	0.90979	0.91082	0.91108
DSR control pkt (c)	9328	9218	9415	9394
Ack pkt (a)	8324	8411	8685	8596
Data pkt (d)	22579	22348	22849	22492
Overhead ratio	0.78178	0.78884	0.79215	0.79983
Power consumption	9259	10183	10776	12846

7. When the traffic pattern is changing then the results looks like that:

Observed parameters	For a traffic pattern of 5 packets/sec.	For a traffic pattern of 5 packets/sec.	For a traffic pattern of 5 packets/sec.
No. of pkt received (r)	2241	2555	3143
No. of pkt sent (s)	2448	2748	3450
Throughput (r/s)	0.91544	0.92976	0.91101
DSR control pkt (c)	9315	9918	10125
Ack pkt (a)	8411	8892	9984
Data pkt (d)	23285	26612	28704
Overhead ratio	0.76126	0.70682	0.70056
Power consumption	10183	11834	12835

8. When the pause time increasing the results looks like that:

Observed parameters	When pause time 0 sec.	When pause time 50 sec.	When pause time 100 sec	When pause time 150 sec.	When pause time 200 sec.
No. of pkt received (r)	2358	2181	2357	2503	2977
No. of pkt sent (s)	2551	2447	2600	2575	3012
Throughput (r/s)	0.92434	0.89129	0.90653	0.97203	0.98837
DSR control pkt (c)	9012	8991	9137	9215	10192
Ack pkt (a)	8117	8274	7999	8172	8215
Data pkt (d)	23737	24323	25414	25907	28665
Overhead ratio	0.72161	0.70982	0.67427	0.67113	0.64214
Power consumption	9460	9820	10840	11200	12501