Study and Implementation on Proactive and Reactive Routing Protocols in MANETs

A

Dissertation

Submitted in partial fulfillment of the requirement
For the award of degree of

MASTER OF ENGINEERING
(Computer Technology and Applications)
(2007-2009)

Submitted by

JAGDEESH PRASAD MEENA
(University Roll No.12204)

Under the Expert Guidance of

Mr. MANOJ SETHI
Head of Department, COMPUTER CENTER
Delhi College of Engineering

DEPARTMENT OF COMPUTER ENGINEERING
DELHI COLLEGE OF ENGINEERING
BAWANA ROAD, DELHI
(DELHI UNIVERSITY)
JUNE-2009

CERTIFICATE

DELHI COLLEGE OF ENGINEERING
(Govt. of National Capital Territory of Delhi)
BAWANA ROAD, DELHI - 110042

This is to certify that **JAGDEESH PRASAD MEENA** under **University Roll no: 12204** has carried out his Dissertation work entitled **"STUDY AND IMPLEMENTATION ON PROACTIVE AND REACTIVE ROUTING PROTOCOLS IN MANETS "** as a partial requirement for the award of Master of Engineering degree in Computer Technology and Application by Delhi College of Engineering, Delhi University, New Delhi and completed under my supervision and guidance during the academic session 2008-2009

(Mr.MANOJ SETHI)
HEAD OF COMPUTER CENTER
DELHI COLLEGE OF ENGINEERING
BAWANA ROAD, DELHI - 110042

ACKNOWLEDGEMENT

This work is not a solo endeavor but rather the amalgamate consequence of contribution from various people and sources. Therefore it would be discourteous to present it without acknowledging their valuable guidance.

It is distinct pleasure to express my deep sense of gratitude and indebtedness to my guide **Mr. Manoj Sethi** for their invaluable guidance, encouragement. His continuous inspiration only has made me complete this dissertation. I am also thankful to **Dr. Daya Gupta** Head of computer Department for her timely guidance and help me to update my knowledge upto date. All of them kept on boosting me time to time and for putting an extra ounce of effort to realize this work.

I would also like thank my teachers Mrs. Rajni Jindal, Dr. S. K. Saxena for their support.. I am also thankful to my freinds for their continuous motivation and giving me time to time help required for the project.

Finally, I thank my parents and almighty God for making me capable to purse degree in such a reputed college and to pursue work in dedicated wide scoped research field.

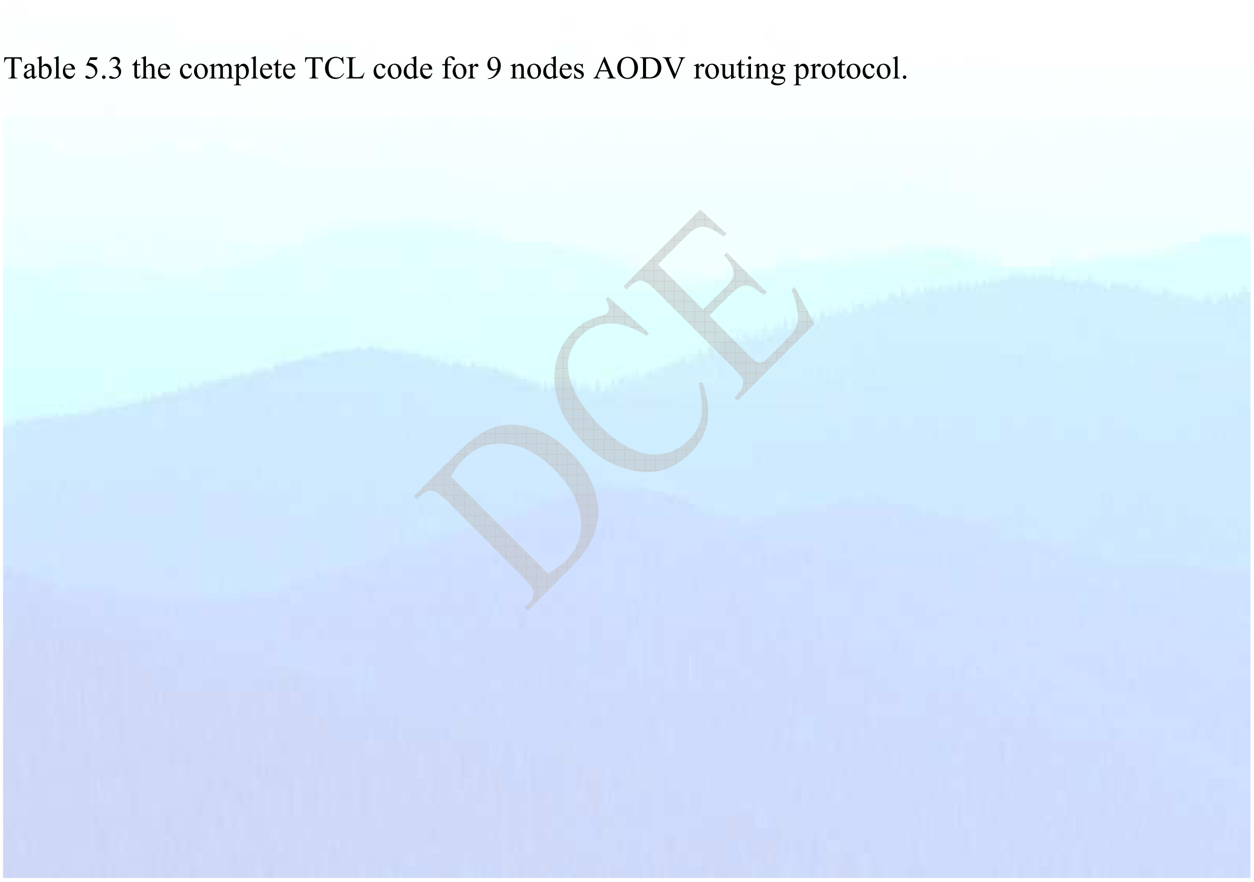
(JAGDEESH PRASAD)
Univ. Roll No. 12204
Class Rollo. 06/CTA/07
Master in Engineering
(Computer Technology & Application)
Dept. of Computer Engineering
DELHI COLLEGE OF ENGINEERING
BAWANA ROAD, DELHI – 110042

Abstract

Ad-hoc networks are characterized by a lack of infrastructure, and by a random and quickly changing network topology; thus the need for a robust dynamic routing protocol that can accommodate such an environment. Consequently, many routing algorithms have come in to existence to satisfy the needs of communications in such networks. This thesis presents a performance comparison between two categories of routing protocols, table-driven (Proactive) and on-demand (Reactive) routing protocols, this two categories were illustrated by using two different examples of routing protocols, first example is DSDV (Destination Sequenced Distance-Vector) from the Proactive family and the second example is AODV (Ad Hoc On-Demand Distance Vector) from the Reactive family. Both protocols were simulated by using NS-2 (network simulator-2) package. Both routing protocols were compared in terms of average throughput (packets delivery ratio) and packet loss ratio, while varying number of nodes and by using the Trace file. Although DSDV perfectly scales to small networks with low node speeds, AODV is preferred due to its more efficient use of bandwidth.

Table of Contents

List of tables



List of figures

1. MANET

1.1 MANET Introduction

Mobile Ad Hoc Network (MANET) is a collection of communication devices or nodes that wish to communicate without any fixed infrastructure and pre-determined organization of available links. The nodes in MANET themselves are responsible for dynamically discovering other nodes to communicate. It is a self-configuring network of mobile nodes connected by wireless links the union of which forms an arbitrary topology. The nodes are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. MANETs are usually set up in situations of emergency for temporary operations or simply if there are no resources to set up elaborate networks. These types of networks operate in the absence of any fixed infrastructure, which makes them easy to deploy, at the same time however, due to the absence of any fixed infrastructure, it becomes difficult to make use of the existing routing techniques for network services, and this poses a number of challenges in ensuring the security of the communication, something that is not easily done as many of the demands of network security conflict with the demands of mobile networks, mainly due to the nature of the mobile devices (e.g. low power consumption, low processing load).

The following flowchart shows the working of any general ad-hoc network

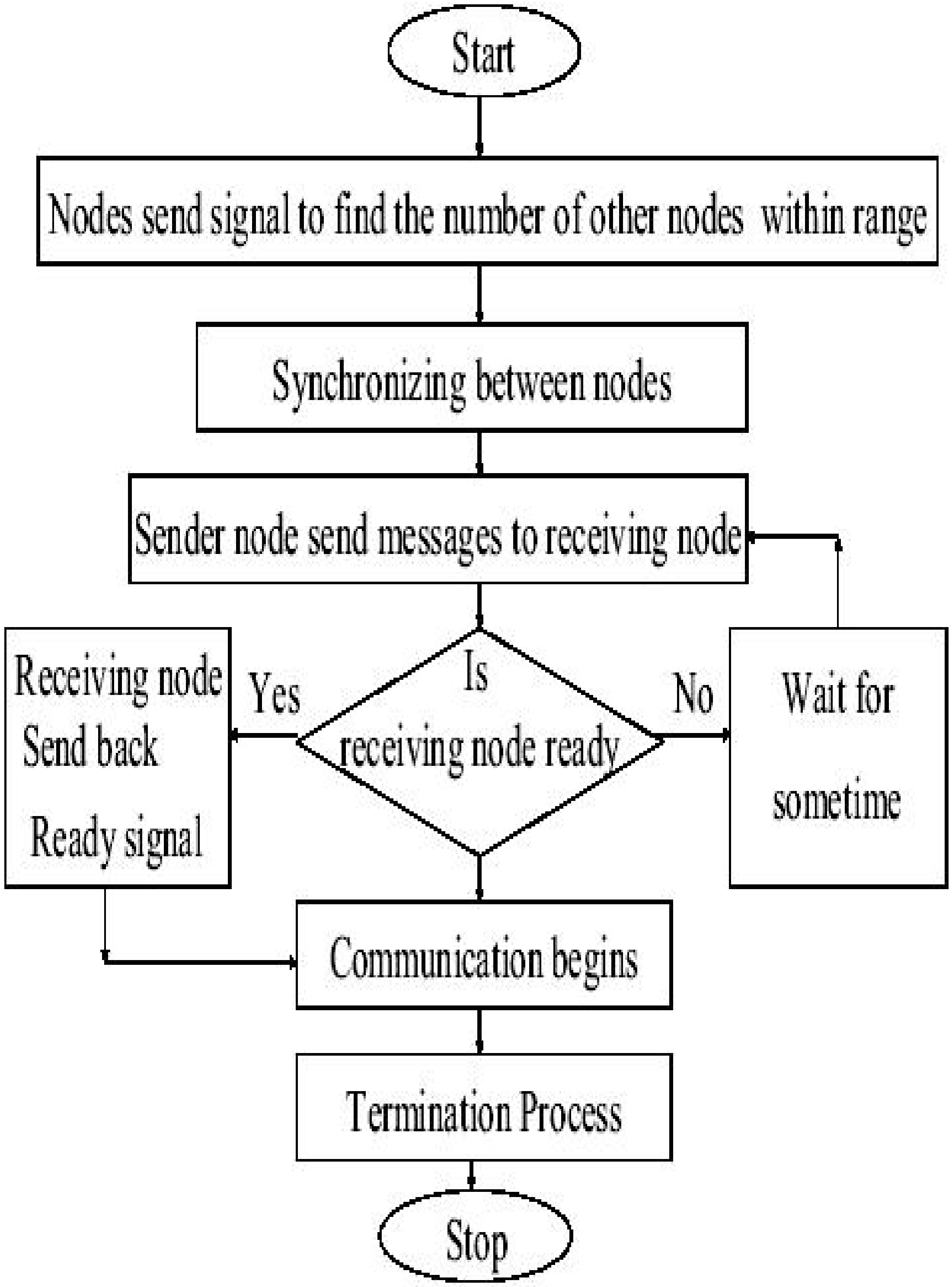

Start
Nodes send signal to find the number of other nodes within range
Synchronizing between nodes
Sender node send messages to receiving node
Is receiving node ready
Yes
No
Receiving node Send back Ready signal
Wait for sometime
Communication begins
Termination Process
Stop

Figure 1.1 working of a general ad-hoc network

1.2 MANET History

The whole life-cycle of ad-hoc networks could be categorized into the first, second, and the third generation ad-hoc networks systems. Present ad-hoc networks systems are considered the third generation.

The first generation goes back to 1972. At that time, they were called PRNET (Packet Radio Networks). The history of ad-hoc networks can be dated back to the DoD1- sponsored Packet Radio Network (PRNET) research for military purpose in 1970s, which evolved into the Survivable Adaptive Radio Networks (SURAN) program in the early 1980s. In conjunction with ALOHA (Arial Locations of Hazardous Atmospheres) and CSMA (Carrier Sense Medium Access), approaches for medium access control and a kind of distance-vector routing PRNET were used on a trial basis to provide different networking capabilities in a combat environment.

The second generation of ad-hoc networks emerged in 1980s, when the ad-hoc network systems were further enhanced and implemented as a part of the SURAN (Survivable Adaptive Radio Networks) program. This provided a packet-switched network to the mobile battlefield in an environment without infrastructure. This program proved to be beneficial in improving the radios' performance by making them smaller, cheaper, and resilient to electronic attacks.

In the 1990s, the concept of commercial ad-hoc networks arrived with notebook computers and other viable communications equipment. At the same time, the idea of a collection of mobile nodes was proposed at several research conferences. Since mid 1990s, a lot of work has been done on the ad hoc standards. Within the IETF, the MANET working group was born, and made effort to standardize routing protocols for ad hoc networks. Meanwhile, the IEEE 802.11 subcommittee standardized a medium access protocol that was based on collision avoidance and tolerated hidden terminals, for building mobile ad hoc network prototypes out of notebooks and 802.11 PCMCIA cards.

There are currently two kinds of Mobile wireless networks:-

The first is known as infrastructure networks with fixed and wired gateways. Typical applications of this type of “one-hop” wireless network include wireless local area networks (WLANs).

The second type of mobile wireless network is the infrastructure less mobile network, commonly known as the MANET. MANET is usually a self-organizing and self configuring "multi-hop" network which does not require any fixed infrastructure. In such network, all nodes are dynamically and arbitrarily located, and are required to relay packets for other nodes in order to deliver data across the network

1.3 Characteristics of MANET

1- Does not rely on a fixed infrastructure for its operation autonomous transitory association of mobile nodes.

2- It can be rapidly deployed with user intervention

3- Need not to operate in a stand alone fashion but can be attached to the Internet or Cellular networks.

4- Devices are free to join or leave the network and they may randomly, possibly resulting in rapid and unpredictable changes.

1.4 Properties of MANET

MANETs have the following special features that should be considered in designing solutions for this kind of networks.

Dynamic Topology: Due to the node mobility, the topology of mobile multi-hop ad hoc networks changes continuously and unpredictably. The link connectivity among the terminals of the network dynamically varies in an arbitrary manner and is based on the proximity of one node to another node. It is also subjected to frequent disconnection during node's mobility. MANET should adapt to the traffic and propagation conditions as well as to the mobility patterns of the mobile network nodes. The mobile nodes in the network dynamically establish routing among themselves as they move about, forming their own network on the fly. Moreover, a user in the

MANET may not only operate within the ad hoc network, but may require access to a public fixed network.

Bandwidth: MANETs have significantly lower bandwidth capacity in comparison with fixed networks. The used air interface has higher bit error rates, which aggravates the expected link quality. Current technologies suitable for the realization of MANETs are IEEE 802.11 with bandwidth up to 54Mbps and Bluetooth providing bandwidth of 1Mbps. The nature of high bit-error rates of wireless connection might be more profound in a MANET. One end-to-end path can be shared by several sessions. The channel over which the terminals communicate is subjected to noise, fading and interference, and has less bandwidth than a wired network. In some scenarios, the path between any pair of users can traverse multiple wireless links and the links themselves can be heterogeneous.

Energy: All mobile devices will get their energy from batteries, which is a scarce resource. Therefore the energy conservation plays an important role in MANETs. This important resource has to be used very efficiently. One of the most important system design criteria for optimization may be energy conservation.

Security: The nodes and the information in MANETs are exposed to the same threats like in other networks. Additionally to these classical threats, in MANETs there are special threats, e.g. denial of service attacks. Also mobility implies higher security risks than static operation because portable devices may be stolen or their traffic may insecurely cross wireless links. Eavesdropping, spoofing and denial of service attacks should be considered.

Autonomous: No centralized administration entity is required to manage the operation of the different mobile nodes. In MANET, each mobile terminal is an autonomous node, which may function as both a host and a router. So usually endpoints and switches are indistinguishable in

MANET.

Distributed Operation: Since there is no background network for the central control of the network operations, the control and management of the network is distributed among the terminals, The nodes involved in a MANET should collaborate among themselves and each node acts as a relay as needed, to implement functions e.g. security and routing.

Multi-hop Routing: Basic types of ad hoc routing algorithms can be single-hop and multi-hop, based on different link layer attributes and routing protocols. Single-hop MANET is simple in comparison with multi-hop MANET in terms of structures and implementation. When delivering data packets from a source to its destination out of the direct wireless transmission range, the packets should be forwarded via one or more intermediate nodes.

Light-Weight Terminals: In most cases, the MANET nodes are mobile devices with less CPU processing capability, small memory size and low power storage.

Infrastructure-less and Self Operated: A mobile ad hoc network includes several advantages over traditional wireless networks, including: ease of deployment, speed of deployment and decreased dependence on a fixed infrastructure. MANET is attractive because it provides an instant network formation without the presence of fixed base stations and system administrators.

1.5 Challenges of Ad-hoc Networks

Ad hoc engineering has to solve two main problems:

- Routing: the mobility of the nodes leads to a more complex task for routing. Indeed, in most of the cases, nodes will move frequently. The changes of topology imply that a dynamic routing protocol needs to maintain routes between source and destination.

- Mobility Management: the routing algorithm has to manage the mobility of the nodes which move randomly and unpredictably in the network. In order to achieve this of task, nodes might store information concerning the topology of the network in their routing protocol table. Hence, the success of the ad hoc network depends of both the quality of the information collected by the node and the efficiency of the routing protocol.
- There are other aspects of the ad hoc networking which could be highlighted like security where trust relationships have to be set up (use of cryptography). Indeed the use of multiple hopes could be a problem because, on one hand, it facilitates the interception of the data by an unauthorized person and on the other hand, the intentional interference or unintentional interference due to the fact that several nodes share the same air interface domain.

1.6 MANET Applications

With the increase of portable devices as well as progress in wireless communication, ad hoc networking is gaining importance with the increasing number of widespread applications. Ad hoc networking can be applied anywhere where there is little or no communication infrastructure or the existing infrastructure is expensive or inconvenient to use. Ad hoc networking allows the devices to maintain connections to the network as well as easily adding and removing devices to and from the network. The set of applications for MANETs is diverse, ranging from large-scale, mobile, highly dynamic networks, to small, static networks that are constrained by power sources. Besides the legacy applications that move from traditional infrastructure environment into the ad hoc context, a great deal of new services can and will be generated for the new environment. Typical applications include:

1) Military battlefield. Military equipment now routinely contains some sort of computer equipment. Ad hoc networking would allow the military to take advantage of commonplace network technology to maintain an information network between the soldiers, vehicles, and military information head quarters. The basic techniques of ad hoc network came from this field.

2) Commercial sector. Ad hoc can be used in emergency/rescue operations for disaster relief efforts, e.g. in fire, flood, or earthquake. Emergency rescue operations must take place where non-existing or damaged communications infrastructure and rapid deployment of a communication network is needed. Information is relayed from one rescue team member to another over a small handheld. Other commercial scenarios include e.g. ship-to-ship ad hoc mobile communication, law enforcement, etc.

3) Local level. Ad hoc networks can autonomously link an instant and temporary multimedia network using notebook computers or palmtop computers to spread and share information among participants at an e.g. conference or classroom. Another appropriate local level application might be in home networks where devices can communicate directly to exchange information. Similarly in other civilian environments like taxicab, sports stadium, boat and small aircraft, mobile ad hoc communications will have many applications.

4) Personal Area Network (PAN). Short-range MANET can simplify the intercommunication between various mobile devices (such as a laptop, and a cellular phone). Tedious wired cables are replaced with wireless connections. Such an ad hoc network can also extend the access to the Internet or other networks by mechanisms e.g. Wireless LAN (WLAN), GPRS, and UMTS. The PAN is potentially a promising application field of MANET in the future pervasive computing context.

1.7 Motivation

Routing protocols in Ad Hoc networks has been an active area of research for sometimes now. In an effort to maximize the throughput of Ad Hoc networks, researchers have proposed various routing protocols, each of them takes in to account some of the sources of service impairment in an Ad Hoc networks.

As the node move in and out of range with respect to one another, those that operate as routers, the resulting topology somehow is communicated to all other nodes so the up to date topology information for routing purposes is maintained. In addition, the communication need of the user applications, the limited bandwidth of wireless channels, and the generally hostile transmission characteristics all impose additional constraints on the type, size and frequency of information to be exchanged. Thus ensuring effective routing is one of the greatest challenges for Ad Hoc networking. Significant work has been done on routing in Ad Hoc networks, some of the well known routing strategies are Direct Sequenced Distance Vector **(DSDV)** and Ad hoc on-demand Distance Vector **(AODV)**. The emphasis is on developing the shortest path between the source and the destination, with an attempt to provide a higher degree of route availability. Ad Hoc network have non-deterministic nature of network topology, in this thesis the performance has been studied through extensive simulations using Network Simulator (NS-2).

2. Ad-hoc Routing Protocols

2.1 Routing Protocols Introduction

A routing protocol is needed to send data from one device to another. Whenever the packet is to travel to its destination via several intermediate nodes, routing protocol is needed. This chapter aims to review strategies widely used in routing protocols. Several well known routing protocols are discussed and analyzed. These routing protocols may generally be categorized as:

• Table-driven (Proactive) Routing Protocol

• On-Demand (Reactive) Routing Protocol

• Hybrid Routing Protocol

Despite being designed for the same type of underlying network, the characteristics of each of these protocols are quite distinct. In the Hybrid Routing Protocol**,** each node maintains the network topology information up to m hops, based on routing information update mechanism. The following sections describe the protocols and categorize them according to their characteristics.

2.2 The OSI Layer

An **open system interconnection (OSI) model** is a set of protocols that allows any two different systems to communicate regardless of their underlying architecture. The purpose of the OSI model is to show how to facilitate communication between different systems without requiring changes to the logic of the underlying hardware and software.

The OSI model is not a protocol; it is a model for understanding and designing a network architecture that is flexible, robust and interoperable.

The OSI model is layered framework for the design of network systems that allows communication between all types of computer systems. If consists of seven separate but related layers, each of which defines a part of the process of moving information across a network (see figure 2.1). Understanding the fundamentals of the OSI model provides a solid basis for exploring data communications.

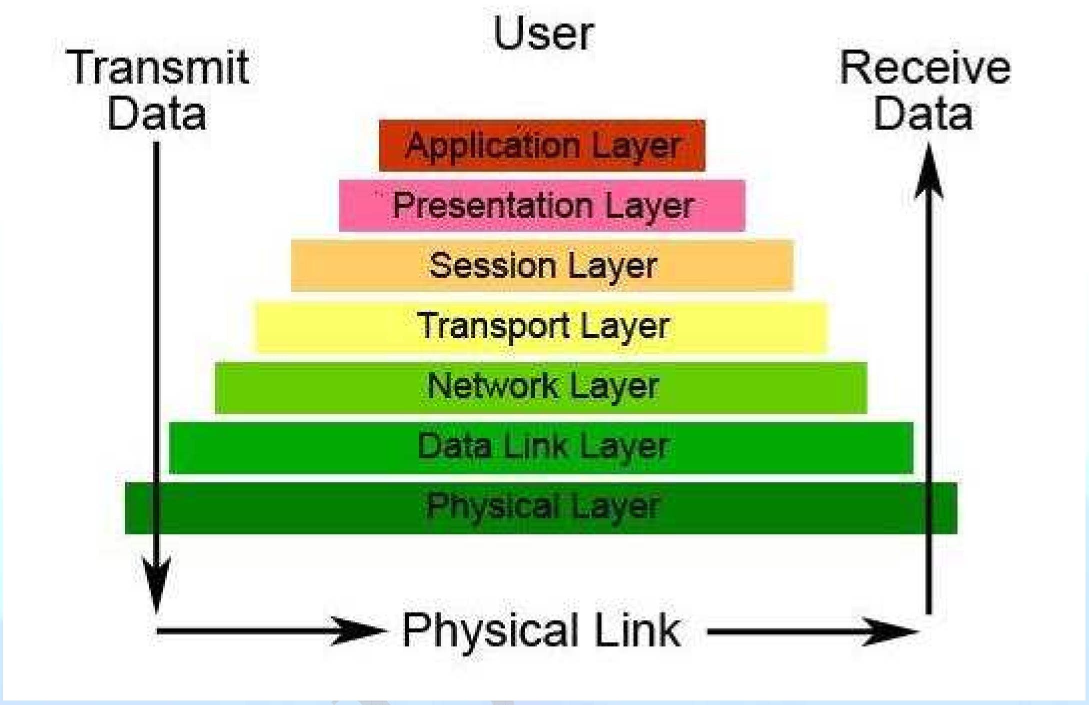


Figure 2.1 the seven layers of OSI

The OSI model is composed of seven ordered layers: physical (layer 1), data link (layer 2), network (layer 3), transport (layer 4), session (layer 5), presentation (layer 6), and application (layer 7). Within a single machine, each layer called upon the services of the layer just below it, For example layer 3 uses the services provided by layer 2 and provide services for layer 4.

Between machines, layer x on one machine communicates with layer x on another machine. This communication is governed by an agreed-upon series of rules and conventions called protocols. The processes on each machine that communicate at a given layer are called (peer-to-peer processes). Communication between machines is therefore a peer-to-peer process using the protocols appropriate to a given layer.

The passing of the data and network information down through the layers of the sending device and back up through the layers of the receiving device is made possible by an interface between each pair of adjacent layers (see figure 2.2). Each interface defines what information and services a layer must provide for the layer above it. Well-define interfaces and layer functions provide modularity to a network. As long as a layer provides the expected services to the layer above it, the specific implementation of its functions can be modified or replaced without requiring changes to the surrounding layers.

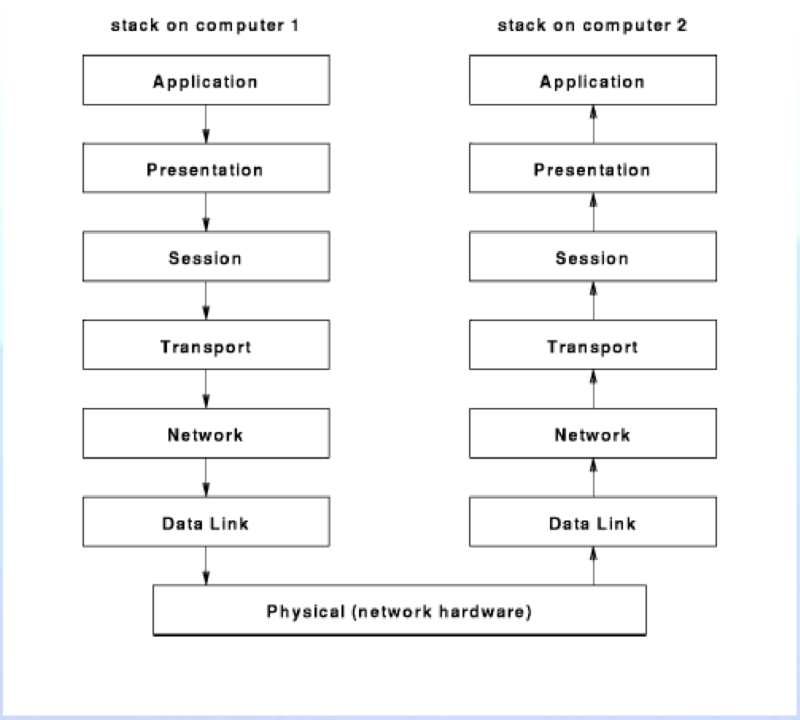


Figure 2.2 the interface between each pair of adjacent layers

2.3 Transmission Control Protocol (TCP)

The TCP/IP protocol suite has specified two protocols for the transport layer: UDP and TCP, Figure 2.3 show the relationship of TCP to the other protocols in the TCP/IP protocol suite.

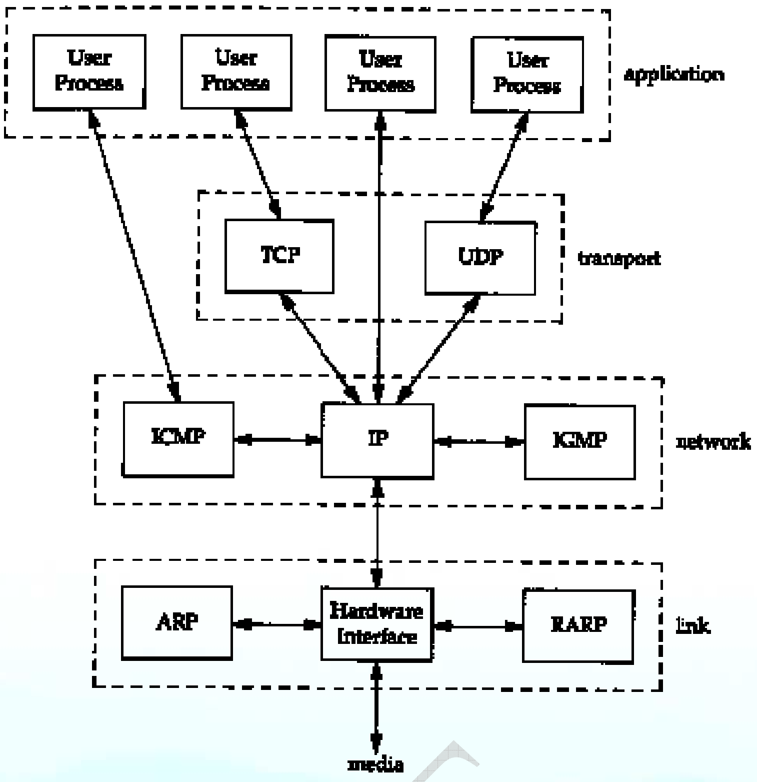


Figure 2.3 TCP/IP protocol suite

TCP lies between the application layer and the network layer and serves as the intermediary between the application programs and the network operations. TCP, unlike UDP, is a process-to-process (program-to-program) protocol. TCP, therefore, like UDP, uses port numbers. Unlike UDP, TCP is a connection-oriented protocol; it creates a virtual connection between two TCPs to send data. In addition, TCP uses flow-and error-control mechanisms at the transport level.

In brief, TCP is called a connection-oriented, reliable transport protocol. It adds connection-oriented and reliability features to the services of IP.

2.3.1 TCP Services

Let us explain the services offered by TCP to the processes at the application layer

- ***Process-to-process Communication:*** Like UDP, TCP provides process-to-process communication using port numbers.
- ***Stream Delivery Service:*** TCP, unlike UDP, is a stream-oriented protocol. It allows the

receiving process to obtain data as a stream of bytes. TCP creates an environment in which the two processes seem to be connected by an imaginary "tube" that carries their data across the Internet.

- ***Sending and Receiving Process:*** because the sending and the receiving process may not write or read at the same speed, TCP needs buffers for storage. There are two buffers, the sending buffer and the receiving buffer, one for each direction.
- ***Full-Duplex Communication:*** TCP offers full-duplex services, where data can flow in both directions at the same time. Each TCP then has a sending and receiving buffer and segments move in both directions.
- ***Connection-Oriented Service:*** TCP, unlike UDP, is a connection-oriented protocol. When a process at site A wants to send a receive data from another process at site B, the following occurs:

  1- The two TCPs establish a connection between them.

  2- Data are exchange in both directions.

  3- The connection is terminated.
- ***Reliable service:*** TCP is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe and sound arrival of data.

2.3.2 TCP Features

To provide the services mentioned in the previous section, TCP has several features that are briefly summarized in this section:

- ***Numbering system:*** although the TCP software keeps track of the segments being transmitted or received, there is no field for a segment number value in the segment header. Instead, there are two fields called the *sequence number* and the *acknowledgment number*. These two fields refer to the byte number and not the segment number.
- ***Flow control:*** TCP, unlike UDP, provides flow control. The receiver of the data controls

how much data are to be sent by the sender. This is done to prevent the receiver from being overwhelmed with data. The numbering system allows TCP to use a byte-oriented flow control.

- ***Error control:*** to provide reliable service, TCP implements an error control mechanism. Although error control considers a segment as the unit of data for error detection (loss or corrupted segments), error control is byte-oriented.
- ***Congestion control:*** TCP, unlike UDP, takes into account congestion in the network. The amount of data sent by a sender is not only controlled by the receiver (flow control), but is also determined by the level of congestion in the network.

2.4 Routed and Routing Protocols

Before explaining the specificity of routing in an ad hoc network a definition of a routed and a routing protocol should be done.

• ***Routed protocols:*** Any network protocol that provides enough information in its network layer address to allow a packet to be forwarded from host to host based on the addressing scheme. Routed protocols are nothing more than data being transported across the networks. They define the format and use of the fields within a packet. Packets generally are conveyed from end system to end system. IP (Internet Protocol), Telnet, RPC (Remote Procedure Call), SNMP (Simple Network Management Protocol) are examples of routed protocols.

• ***Routing protocols****:* Facilitate the exchange of routing information between networks, allowing routers to build routing tables dynamically. Routing protocols are the routes which are available and which are the most efficient routes to a destination. RIP and RIP II (Routing Information Protocol), OSPF (Open Shortest Path First), BGP (Border Gateway Protocol) are examples of routing protocols.

2. 5 Problems Encounters While Routing in an Ad-hoc Network

Routing in a MANET s is an important aspect of a protocol, and will influence on the overall end to end "quality" of the link, each routing protocol has a specific domain and purpose in which it can be used. Below are some aspects that should take into account before defining in which way the protocol should be used. In reference number some of the following parameters are given like:

• ***Network size/scalability:*** as the number of nodes increases, the greater the burdens on a flat routing scheme is, and hence hierarchical routing protocols may be needed for large number of nodes. In addition, some networks (e.g. mobile military networks or highway networks) may be relatively large (e.g. tens or hundreds of nodes per routing area), therefore a scalable protocol should be necessary.

• ***Geographical area:*** how spread out the nodes are, E.g. in an amusement park, where a high density of nodes are available or in an automotive situation where density can be from high during rush hours to low during afternoons.

• ***The topology rate of change:*** this is major performance-impacting parameters for ad hoc routing protocols. For example, the proactive routing protocols are very inefficient with a highly changing topology, since a lot of routes finding is wasted.

• ***QoS (Quality of Service):*** Wireless links will continue to have significantly lower capacity than their hardwired counterparts. In addition, the realized throughput of wireless communications is often much lower than a radio's maximum transmission rate. The aspect of multiple access, fading, noise, and interference conditions, should be taken into account. The route acquisition time is a particular aspect of end-to-end delay and is a special aspect of "on demand" routing algorithms as the time required to establish route(s) is longer than only searching in a existing lookup table.

• ***Energy-constrained operation:*** Some or all of the nodes in a MANET may rely on batteries. For these nodes, the most important system design criteria for optimization may be energy conservation.

• ***Limited physical security:*** Mobile wireless networks are generally more prone to physical security threats than are fixed cable nets. The increased possibility of eavesdropping, spoofing, and denial-of-service attacks should be carefully considered. One, strength of the multi hop networks is the decentralized nature of network control in MANETs which provides additional robustness against the single points of failure of more centralized approaches.

2. 6 Routing Protocol Classification

Many routing protocols exist; they have different properties and working mechanisms. Depending upon their properties routing protocols have been classified in to three categories as:

- Centralized v/s Distributed
- Static v/s Adaptive
- Reactive v/s Proactive

Centralized v/s Distributed: In centralized routing protocol all route choices are made at a central node. This type of routing protocol assumes that the source nodes will determine the entire route; this is usually referred to as source routing. In case of distributed algorithms the computation of routes is shared the network nodes. This protocol assumes that host knows nothing about the routes. In these protocols, routers determine the path through the internet work based on their calculations. In the first system, the hosts have the routing intelligence. In the latter systems, routes have the routing intelligence.

Static v/s Adaptive: This category of routing protocols relates to whether they change routes in response to the traffic input patterns. In static algorithms, the route used by source destination pairs is fixed regardless of traffic conditions. Protocols just see table mappings established by the network administrator before the beginning of routing. These mapping do not change unless the network administrator alters them. Because static routing systems cannot react to network changes, they generally are considered unsuitable for today's large, constantly changing networks. Most of the dominant routing protocols today are dynamic routing protocols, which adjust to changing network circumstances by analyzing incoming routing update messages. If the message indicates that a network change has occurred, the routing software recalculates routes and sends out new routing update messages.

Proactive V/s Reactive: this category of protocols is more relevant to ad hoc networks. Reactive protocols maintain routes between nodes that need to communicate. Proactive protocols maintain routes between all node-pairs. Proactive Protocols are also referred to as Table Driven Routing protocols and reactive protocols are also called On Demand Routing Protocols. Next section explains these two protocols in details.

3. Proactive and Reactive Routing Protocol

3.1 Proactive Routing Protocol

These protocols are also referred to as Table Driven Routing Protocols. These protocols are extensions of the wired network routing protocols. They maintain the global topology information in the form of tables at every node. These tables are updated frequently in order to maintain consistent and accurate network state information. In proactive routing protocols, nodes continuously search for routing information with in a network, so that when a route is needed, the route is already known. Periodically floods the network to reconstruct the routing table E.g. DSDV (Destination Sequenced Distance Vector) routing protocol. The Destination Sequenced Distance- Vector (DSDV) routing protocol, is example for the protocols that belong to this category.

Their main characteristics are:

- Low route latency
- High overhead (periodic table updates)
- Route repair depends on update frequency.

3.1.1 Destination Sequenced Distance Vector (DSDV) Routing Protocol

The Destination Sequenced Distance Vector Routing protocol (DSDV) is a table-driven algorithm based on the classical Bellman-Ford routing mechanism. The improvements made to the Bellman-Ford algorithm include freedom from loops in routing tables.

Every mobile node in the network maintains a routing table in which all of the possible destinations within the network and the number of hops to each destination are recorded. Each entry is marked with a sequence number assigned by the destination node.

The sequence numbers enable the mobile nodes to distinguish stale routes from new ones, thereby avoiding the formation of routing loops. Routing table updates are periodically transmitted throughout the network in order to maintain table consistency. To help alleviate the potentially large amount of network traffic that such updates can generate, route updates can employ two possible types of packets. The first is known as a full dump. This type of packet carries all available routing information and can require multiple network protocol data units (NPDUs). During periods of occasional movement, these packets are transmitted infrequently. Smaller incremental packets are used to relay only that information which has changed since the last full dump. Each of these broadcasts should fit into a standard-size NPDU, thereby decreasing the amount of traffic generated. The mobile nodes maintain an additional table where they store the data sent in the incremental routing information packets.

New route broadcasts contain the address of the destination, the number of hops to reach the destination, the sequence number of the information received regarding the destination, as well as a new sequence number unique to the broadcast. The route labeled with the most recent sequence number is always used. In the event that two updates have the same sequence number, the route with the smaller metric is used in order to optimize (shorten) the path. Mobiles also keep track of the settling time of routes, or the weighted average time that routes to a destination will fluctuate before the route with the best metric is received. By delaying the broadcast of a routing update by the length of the settling time, mobiles can reduce network traffic and optimize routes by eliminating those broadcasts that would occur if a better route was discovered in the very near future.

3.1.2 Advantages and Disadvantages

The availability of routes to all destinations at all times implies that much less delay is involved in the route setup process. The mechanism of incremental updates with sequence number tags makes the existing wired network protocols adaptable to ad hoc wireless networks.

Hence, an existing wired network protocol can be applied to ad hoc wireless networks with many fewer modifications. The updates are propagated throughout the network in order to maintain an up-to-date view of the network topology at all the nodes. The updates due to broken links lead to a heavy control overhead during high mobility. Even a small network with high mobility or a large network with low mobility can completely choke the available bandwidth. Hence, this protocol suffers from excessive control overhead that is proportional to the number of nodes in the network and therefore is not scalable in ad hoc wireless networks, which have limited bandwidth and whose topologies are highly dynamic.

The protocol requires selection of the following parameters: periodic update interval, maximum value of the settling time for a destination and the number of update intervals which may transpire before a route is considered stale. These parameters will likely represent a tradeoff between the latency of valid routing information and excessive communication overhead. Another disadvantage of DSDV is that in order to obtain information about a particular destination node, a node has to wait for a table update message initiated by the same destination node. This delay could result in stale routing information at nodes.

3.2 Reactive Routing Protocol

These protocols are also referred to as On Demand Driven or the source initiated routing protocol. It is the second category under ad hoc mobile routing protocols. For these types of protocols, it creates routes only when desired by source nodes. When a node requires a route to destination, it initiates route discovery process within the network. This process completes once one route is found or all possible route permutations are examined. Once a route is discovered and established, it is maintained by route maintenance procedure until either destination becomes inaccessible along every path from source or route is no longer desired.

Their main characteristics are:

- High route latency

- No overhead from periodic update
- Route changing can reduce latency.

3.2.1 Ad-hoc On-Demand Distance Vector (AODV) Routing Protocol

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

When a source node desires to send a message to some destination node and does not already have a valid route to that destination, it initiates a path discovery process to locate the other node.

It broadcasts a route request (RREQ) packet 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. (Figure 3.1 a) illustrates the propagation of the broadcast RREQs across the network. AODV utilizes **destination sequence numbers** to ensure all routes are loop-free and contain the most recent route information. Each node maintains its own sequence number, as well as a broadcast ID. The broadcast ID is incremented for every RREQ the node initiates, and together with the node's IP address, uniquely identifies an RREQ. Along with its own sequence number and the broadcast ID, the source node includes in the RREQ the most recent sequence number it has for the destination. Intermediate nodes can reply to the RREQ only if they have a route to the destination whose corresponding destination sequence number is greater than or equal to that contained in the RREQ.

During the process of forwarding the RREQ, intermediate nodes record in their route tables the address of the neighbor from which the first copy of the broadcast packet is received, thereby establishing a reverse path. If additional copies of the same RREQ are later received, these packets are discarded. Once the RREQ reaches the destination or an intermediate node with a fresh enough route, the destination/intermediate node responds by unicasting a route reply (RREP) packet back to the neighbor from which it first received the RREQ (Fig. 3.1 b). As the RREP is routed back along the reverse path, nodes along this path set up forward route entries in their route tables which point to the node from which the RREP came. These forward route entries indicate the active forward route.

Associated with each route entry is a route timer which will cause the deletion of the entry if it is not used within the specified lifetime. Because the RREP is forwarded along the path established by the RREQ, AODV only supports the use of symmetric links.

Routes are maintained as follows. If a source node moves, it is able to reinitiate the route discovery protocol to find a new route to the destination. If a node along the route moves, its upstream neighbor notices the move and propagates a link failure notification message (an RREP with infinite metric) to each of its active upstream neighbors to inform them of the erasure of that part of the route.

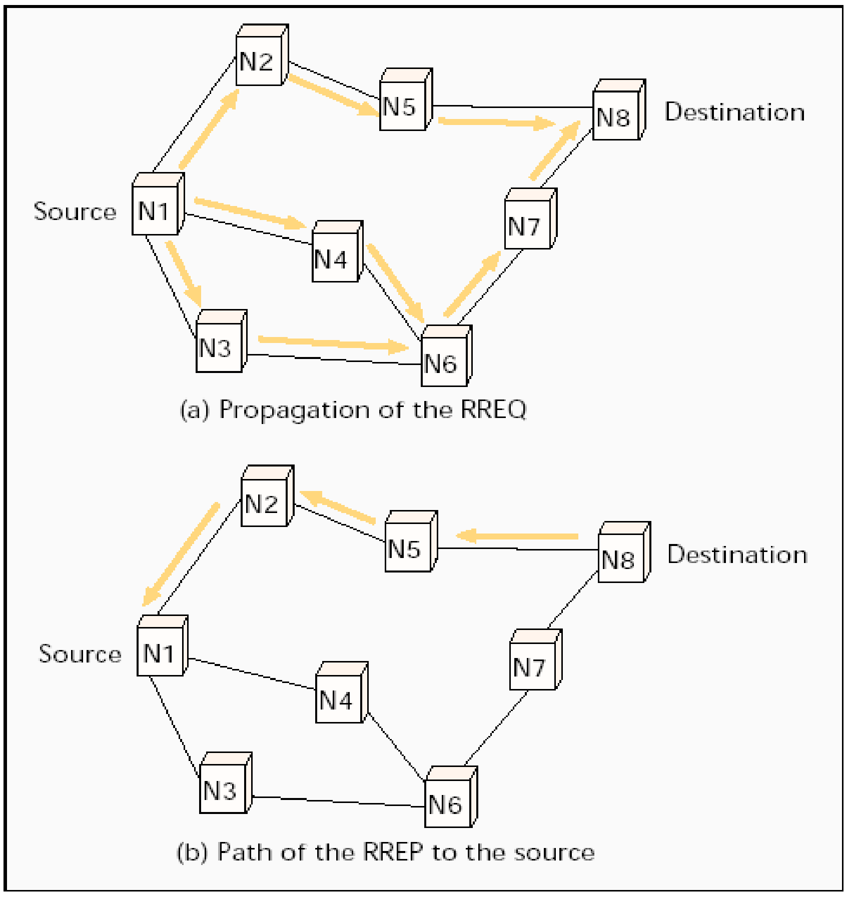


Figure 3.1 AODV route discoveries

These nodes in turn propagate the link failure notification to their upstream neighbors, and so on until the source node is reached.

An additional aspect of the protocol is the use of hello messages, periodic local broadcasts by a node to inform each mobile node of other nodes in its neighborhood. Hello messages can be used to maintain the local connectivity of a node. However, the use of hello messages is not required. Nodes listen for retransmission of data packets to ensure that the next hop is still within reach. If such a retransmission is not heard, the node may use any one of a number of techniques, including the reception of hello messages, to determine whether the next hop is within communication range. The hello messages may list the other nodes from which a mobile has heard, thereby yielding greater knowledge of network connectivity.

3.2.2 Advantages and Disadvantages

The main advantage of this protocol is that, routes are established on demand and destination sequence numbers are used to find the latest route to the destination and the connection setup delay is less. Also its uses bandwidth efficiently by minimizing the network load for control and data traffic, it is repetitive to change in topology, it is scalable and ensures loop free routing.

One of the disadvantages of this protocol is that intermediate nodes can lead to inconsistent routes if the source sequence number is very old and the intermediate nodes have a higher but not the latest destination sequence number, thereby having stale entries. Nodes use the routing cashes to reply to route queries, this result in an uncontrolled replies and repetitive updates in hosts cashes yet early queries cannot stop the propagation of all query messages which are flooded all over the network.

Besides that AODV uses periodic beaconing to keep routing tables updated and this creates a significant overhead to the protocol. Also multiple RouteReply packets in response to a single RouteRequest packet can lead to heavy control overhead

3.3 Comparison between On-Demand and Table-Driven Protocols

As discussed earlier, the proactive routing protocol approach is similar to the connectionless approach of forwarding packets, with no regard to when and how frequently such routes are desired. It relies on an underlying routing table update mechanism that involves the constant propagation of routing information. This is not the case; however, for reactive routing protocols. When a node using on demand protocol desires a route to a new destination, it will have to wait until such a route can be discovered. On the other hand, because routing information is constantly propagated and maintained in table-driven routing protocols, a route to every other node in the ad hoc network is always available, regardless of whether or not it is needed. This feature, although useful for datagram traffic, incurs substantial signaling traffic and power consumption. Since both bandwidth and battery power are scarce resources in mobile computers, this becomes a serious limitation.

These two types of protocols have their own working areas. At some places one type is suitable and in others second category is used. Choices of protocol depend on the type of network in operation and working requirements. Time complexity is defined as the number of steps needed to perform a protocol operation, and communication complexity is the number of messages needed to perform a protocol operation. Also, the values for these metrics represent worst case behavior.

Some of the differences between proactive (Table-driven) and reactive (On-demand) routing protocols are shown in Table 3.1.

Parameters	**Proactive-table driven**	**Reactive- on demand**
Overhead	High	Low
Memory requirement	High	Low
Delay	Low	High
Routing philosophy	flat	flat
Availability of routing information	Always available regardless of need	Available when need it
Coping with mobility	Inform other nodes to achieve a consistent routing table	Use localized route discovery
Quality of service (QoS) support	Mainly shortest path as the Qos metric	Few can support QoS, although most support shortest path
Signaling traffic generated	Greater than that in on-demand routing	Grows with increasing mobility of active routes

Table 3.1 Overall comparisons of proactive and reactive routing protocols

4. Network Simulator (NS-2)

4.1 Network Simulator Introduction

Network simulator 2 (NS-2) is an open source discrete event simulation tool used for simulating Internet protocol (IP) networks. It was developed by UC Berkeley and widely used worldwide for network simulation purposes. The NS-2 software uses TCL as a front-end interpreter and C++ as the back end network simulation engine. Network simulation scripts in TCL are used to create the network scenarios and upon the completion of the simulation, trace files that capture events occurring in the network are produced. The trace files would capture information that could be used in performance study, e.g. the amount of packets transferred from source to destination, the delay in packets, packet loss etc. However, the trace file is just a block of ASCII data in a file and quite cumbersome to access using some form of post processing technique.

In order to ease the process of extracting data for performance study, the NS-2 Trace Analyzer is proposed. This software is a tool for extracting and presenting trace files for the network simulation environment of NS-2. The NS-2 Trace Analyzer software consists of three layers. The first layer is the source layer which consists of the trace file data. The second layer is the processing layer. This layer processes the data obtain from the source and convert it to meaningful format for the third layer. The third layer is the presentation layer. This layer presents meaningful data in the form of graph, table and report for network performance study, i.e. throughput, end-to-end delay, packet loss.

Through the NS-2 Trace Analyzer the user would be able to do performance study of a network scenario through interactive GUI. This will benefit the user since he or she can concentrate on developing new algorithms or new architectures rather spending too much time on post processing of data.

NS2 undoubtedly is a very powerful network simulator that has been used widely. However, unfortunately it does not provide tools to represent results. To simulate a certain network scenario, NS-2 users need to write a simulation script, save it and invoke the NS2 interpreter. After that NS-2 will simply store the results in form of trace files. There are two types of trace files.

This means that the NS-2 user will end up creating their own program to process the trace files to represent it in a presentable form. One popular approach is to produce two types of trace file, i.e. network animation (NAM) trace file and normal trace file. NAM trace file is used for network animation purposes. While for trace file processing, a program can be coded using the user's own favorite software and graph plotters like GNU plot and x-graph can be used to view the results. Fig.4.1 shows the approach taken.

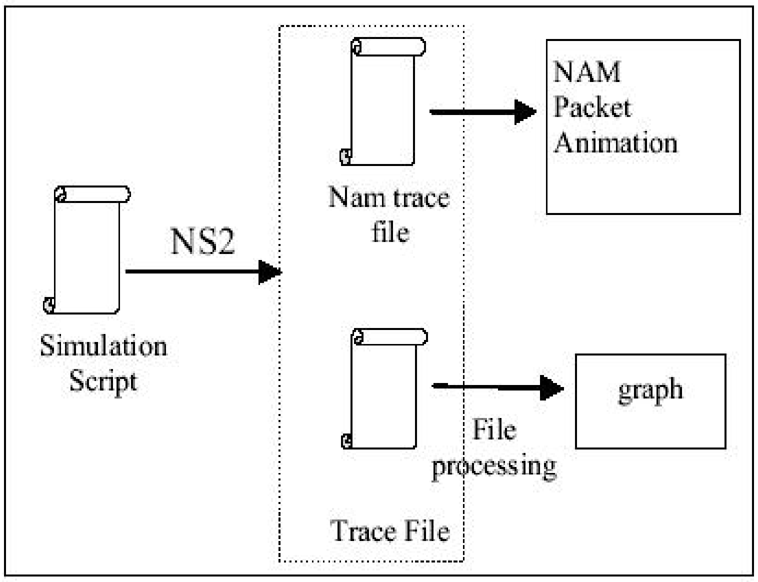


Figure 4.1 Ns-2 Simulation Process Flow

4.2 NS-2 Trace File

```
r 55.032257812 _1_ AGT  --- 25 tcp 1060 [13a 1 2 800] ------- [0:0 1:0 31 1] [1 0] 2 0
r 55.056807709 _1_ AGT  --- 27 tcp 1060 [13a 1 2 800] ------- [0:0 1:0 31 1] [2 0] 2 0
r 55.091395573 _1_ AGT  --- 28 tcp 1060 [13a 1 2 800] ------- [0:0 1:0 31 1] [3 0] 2 0
```

```
r 55.103810505 _1_ AGT  --- 30 tcp 1060 [13a 1 2 800] ------- [0:0 1:0 31 1] [4 0] 2 0
r 55.130038361 _1_ AGT  --- 33 tcp 1060 [13a 1 2 800] ------- [0:0 1:0 31 1] [5 0] 2 0
r 55.164636110 _1_ AGT  --- 34 tcp 1060 [13a 1 2 800] ------- [0:0 1:0 31 1] [6 0] 2 0
r 55.187018902 _1_ AGT  --- 35 tcp 1060 [13a 1 2 800] ------- [0:0 1:0 31 1] [7 0] 2 0
r 55.198894818 _1_ AGT  --- 37 tcp 1060 [13a 1 2 800] ------- [0:0 1:0 31 1] [8 0] 2 0
r 55.233752555 _1_ AGT  --- 40 tcp 1060 [13a 1 2 800] ------- [0:0 1:0 31 1] [9 0] 2 0
```

Table 4.1 a part from the trace file

The trace data is in ASCII code and are organized in 12 fields as shown in Figure above,

- The first field is a letter that can have the values r for "received", s for "sent", f for "forwarded", D for "dropped". It can also be M for giving a location or a movement indication.
- The second field is the simulation time (in seconds) of that event.
- The third field is the node number.
- The fourth field is MAC to indicate if the packet concerns a MAN layer; it is AGT to indicate a transport layer (e.g. tcp) packet, or RTR if it concerns the routed packet. It can also be IFQ to indicate events related to the interference priority queue (like drop of packet).
- After the dashes, come the global sequence number of the packet (this is not the tcp sequence number).
- The next field comes more information on the packet type (tcp, ack or udp).
- Then, in the next field comes the packet size in bytes.
- The 4 numbers in the first square brackets concern (MAC layer information). The first hexadecimal number, 13a specifies the expected time in seconds to send this data packet over the wireless channel. The second number, 1, stands for the MAC-id of the sending node, and the third, 2, is that of the receiving node. The fourth number, 800, specifies that

the MAC type.

- The next numbers in the second square brackets concern the IP source and destination addresses, then the TTL (Time To Live) of the packet
- The third brackets concern the tcp information: its sequence number and the acknowledgement number.
- And the last field shows the unique id of the packet

4.3 NS-2 Languages

The ns simulator covers a very large number of applications, protocols, network types, network elements and traffic models. We call these "simulated objects ".

NS simulator is based on two languages an object oriented simulator, written in C++, and an OTCL (an object oriented extension of TCL) interpreter, used to execute user's command scripts.

NS has a rich library of network and protocol objects. There are two class hierarchies: the compiled C++ hierarchy and the interpreted OTCL one, with one to one correspondence between them.

The compiled C++ hierarchy allows us to achieve efficiency in the simulation and faster execution times. This is in particular useful for the detailed and operation of protocols. This allows one to reduce packet and event processing time.

Then in the OTCL script provided by the user, we can define a particular network topology, the specific protocols and applications that we wish to simulate (whose behavior is already defined in the compiled hierarchy) and the form of the output that we wish to obtain from the simulator. The OTCL can make use of the objects compiled in C++ through an OTCL linkage, which creates a matching of OTCL object for each of the C++.

NS is a discrete event simulator, where the advance of time depends on the timing of events which are maintained by a scheduler. An event is an object in the C++ hierarchy with a unique ID, a schedule time and the pointer to an object that handles the event. The scheduler keeps an ordered data structure with the events to be executed and fires them one by one, invoking the handler of the event.

4.4 Tool Command Language

The TCL (***Tool Command Language***) is a language with a simple syntax and it allows a very easy integration with other languages, the characteristics of this language is:

- It allows a fast development
- It provide a graphical interface
- It is compatible with many platforms
- It is flexible for integration
- It is easy to use
- It is free

The main reason for using a TCL language in NS-2 is because it is not useful to use only C++ code. In fact, by this mean, the user does not need to compile the simulator every time he wants to do a new simulation. The TCL language is interpreted by the C++ code in NS-2, without being compiled.

To use a network simulator, you have to define two things; the first thing is you have to define is, how does the protocol behave? It is done by the C++ code, because it does not change for every simulation. The second thing which you have to define is what are the simulation parameters? It is done by the TCL code, because every simulation is different (number of nodes, positions, protocol used).

The TCL code allows the user to choose between fix or wireless network, and among the different implemented protocols: DSDV and AODV (for wireless networks). The TCL file contains also information's about nodes like position and speed, or information's about source and destination, the transmission rate, and a lot of other parameters. The syntax of this language is defined in the NS-2 manual. Some tools have been developed to build these scenarios.

Like C++, TCL is an object language. So, there are parallels between C++ objects and the TCL objects. A C++ object can be used in the TCL language. If the user needs to share a C++ object with the TCL code, he/she needs to use the TCL object. This class is developed independently to NS-2. It allows defining the TCL name of the C++ object, and then the C++ object is used in the TCL file, using this name.

The TCL object communicates with its corresponding C++ object by using some basic commands. These commands are defined in the commands () function of the C++ commands, which takes the known parameters argc and argv. Thus, the TCL object can initialize the C++ object.

For instance, some C++ objects represent the different layers of the different nodes in the simulation. So, when a packet is sent from one node to another one, the packet goes through different C++ objects.

In the wireless layer, these packets are received in the recv function, if the node is the destination and in the tap function if the node is a neighbor and can listen to the packet even if it is not the destination.

The simulation runs with a specific simulation time, not the real one. Then, by sending packets, C++ objects creates some events: they want that, one C++ object receive one packet at a time, by introduction a delay for example, but because NS-2 is a mono process program, there is a table, containing all the events and sorting them according to the time they occur.

There are two kinds of interaction with the C++ code:

- The first one is with TCL files, which can describe the initial conditions. It can also describe some events, like the change of speed.

- The second one is the generation of events by the C++ code itself.

The C++ code produces some output files, which contain different types of results from the simulation. one can be used to have a graphic display of the simulation using tool NAM. Another one contains trace of all the packets, and can be analyzed with other tools to have statistics about the simulation.

5. Mobile Networks

5.1 Mobile Networks Introduction

There are two approaches for wireless communication between two hosts. The first is the centralized cellular network in which each mobile is connected to one or more fixed base stations (each base station is responsible for another cell), so that a communication between two mobile stations require to involve one or more base stations. A second decentralized approach consists based of an ad-hoc network between users that wish to communicate between each other. Due to the more limited range of a mobile terminal (with respect to a fixed base station), the approach requires mobile nodes not only to be sources or destination of packets but also to forward packets between other mobiles. Cellular station has a much larger range than ad-hoc networks. However, ad-hoc networks have the advantage of being quickly deployable as they do not require an existing infrastructure.

In cellular networks, the wireless part is restricted only to the access to a network, and within the network classical routing protocols can be used. Ad-hoc network in contrast rely on special routing protocols that have to be adapted to frequent topology changes. To model well cellular networks, we will use the simulation tools of NS-2 (Network Simulator) with Linux operating system.

In ad-hoc networks, the routing protocols are central. Na allows simulating the main existing routing as well as transport and applications that use them. Moreover, it allows taking in o account the MAC and link layer, the mobility, and some basic features of the physical layer.

The current routing protocols, which we implemented by using NS-2, are:

- DSDV – Destination Sequenced Distance Vector.
- AODV – Ad-hoc On Demand Distance Vector.

5.2 The Routing Algorithms

There are several approaches in conventional algorithms in traditional wire line networks, and some ideas from these are also used in ad-hoc networks. Among the traditional approaches we shall mention the following:

1- ***Link State:*** Each node maintain a view of the complete topology with a cost per each link, each node periodically broadcasts the link costs of its outgoing links to all other nodes using flooding. Each node updates its view of the network and applies a shortest path algorithm for choosing the next-hop for each destination.

2- ***Distance Vector:*** Each node only monitors the cost of its outgoing links. Instead of broadcasting the information to all nodes, it periodically broadcasts to each of its neighbors an estimate of the shortest distance to every other node in the network. The receiving nodes use this information to recalculate routing tables using path algorithm. This method is more computation efficient, easier to implement and requires less storage space than link state routing.

3- ***Source Routing:*** Routing decisions are taken at the source and packets carry along the complete path they should take.

4- ***Flooding:*** The source sends the information to all neighbors who continue to sending it to their neighbors etc. by using sequence numbers for the packets; a node is able to relay a packet only once.

5.2.1. Destination Sequenced Distance Vector (DSDV)

DSDV is a distance vector routing protocol. Each node has a routing table that indicates for each destination, which is the next hop and number of hops to the destination. Each node periodically broadcasts routing updates. A sequence number is used to tag each route. It shows the freshness of the route, a route with higher sequence number is more favorable. In addition, among two routes with the same sequence number, the one with fewer hops is more favorable. If a node detects that a route to a destination has broken, then its hop number is set to infinity and its sequence number updated (increased) but assigned an odd number, even numbers correspond to sequence numbers of connected paths.

5.2.2. Ad-hoc On-Demand Distance Vector (AODV)

AODV is a distance vector type routing. It does not require nodes to maintain routes to destinations that are not actively used. As long as the endpoints of a communication connection have valid routes to each other, AODV does not play a role. The protocol uses different to discover and maintain links, Route Requests (RREQs), Route Replies (RREPs), and Route Errors (RERRs). These message types are received via TCP, UDP, and normal IP header processing applies.

AODV uses a **destination sequence number** for each route entry. The destination sequence number is created by the destination for any route information, it sends to requesting nodes. Using destination sequence numbers ensures loop freedom and allows which of several routes is more "fresh". Given the choice between two routes to a destination, a requesting node always selects the one with the greatest sequence number.

When a node wants to find a route to another one, it broadcasts a RREQ to all the network till either the destination is reached or another node is found with a "fresh enough" route to the destination (a "fresh enough" route is a valid route entry for the destination whose associated sequence number is al least as great as that contained in the RREG). Then a RREP is sent back to the source and the discovered route is made available.

Nodes that are part of an active route may offer connectivity information by broadcasting periodically local Hello messages (special RREP message) to its immediate neighbors. If Hello messages stop arriving from a neighbor beyond some given time threshold, the connection is assumed to be lost.

When a node detects that a route to a neighbor node is not valid it removes the routing entry and sends a RERR message to neighbors that are active and use the route, this is possible by maintaining active neighbor lists. This procedure is repeated at nodes that receive RERR messages. A source that receives an RERR can reinitiate a RREQ message; AODV does not allow handling unidirectional links.

5.3 Simulation Scenario for DSDV Routing Protocol

We start by presenting simple script that runs a single TCP connection with 6-nodes network over an area of a size of 500m over 400m depicted in fig 5.1.

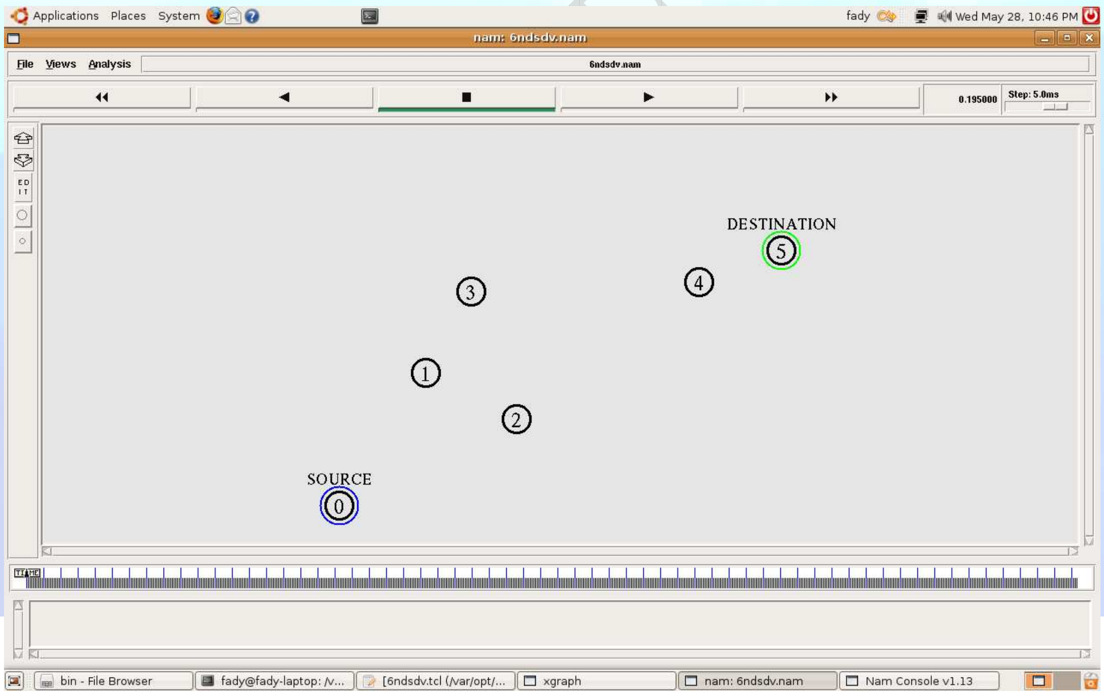


Figure 5.1 topology of a six nodes ad-hoc network

The initial location of nodes 0, 1, 2, 3, 4, 5 are respectively (5, 5), (100,150), (200,100), (150,240), (400,250), (490,285), the Z coordinate is assumed throughout to be 0.

The simulation lasts at 200 second, $ns at 200.0 "finish". At time 10, a TCP connection is initiated between node 0 and node 5, with using DSDV routing protocol.

5.3.1. TCL Script

We begin by specifying some basic parameters for the simulations, providing information for the different layers. This is done as follows:

```
# 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 val(ifqlen)         50                         ;# max packet in
                                                   ifq
set val(nn)             6                          ;# number of
                                                   mobile nodes
set val(rp)             DSDV                       ;# routing
                                                   protocol
set val(x)              500                        ;# X dimension of
                                                   topography
set val(y)              400                        ;# Y dimension of
                                                   topography
set val(stop)           150                ;# time of simulation end
```

Initialization and Termination

An NS simulation starts with the command: set ns [new Simulator], which is used to declares a new variable NS using the set command, you can call this variable as you wish, but in general people declares it as NS because it is an instance of the simulator class, so an object. The code [new Simulator] is indeed the instantiation of the class simulator using the reserved word new. So, using this new variable NS we can use all the methods of the class simulator.

In order to have output files with data on the simulation (trace files) or files used for visualization (NAM files), we need to create the files using the "open" command:

```
#open the trace file
set tracefd      [open "| grep \"r\" | grep \"_5_\" | grep \"tcp\" >
tracefile2.tr" w]
$ns trace-all $tracefd

#open the NAM trace file
set namtrace      [open 6ndsdv.nam w]
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

set windowVsTime2 [open 6ndsdv.tr w]
```

The above creates a data trace file called "tracefile2.tr" and a NAM visualization trace file (for the NAM tool) called "6ndsdv.nam". Within the TCL script, these files are not called explicitly by their names, but instead by pointer that are declared above and called "tracefd" and "namtrace" respectively.

The first and fifth lines in the code above are only comments, they are not simulation commands, remark that they begins with a # symbol. The second line open the file "tracefile2.tr" to be used for writing, declared with the letter "w". The fourth line uses a simulator method called "trace-all" that have as parameter the name of the file where the trace will go. With this simulator command we will trace all the events in a specific format, the last line tells the simulator to record all simulation traces in NAM input format. It also gives the file name that the trace will be written to later by the command $ns flush-trace (see procedure "finish" below) after that we are using this command set windowVsTime2 [open 6ndsdv.tr w] to convert the events in the trace file to diagram and it will draw the window size.

.

The commands trace-all and namtrace-all may result in the creation of huge files. If we wish to save space, other trace commands should be used so as to create traces only a subset of the simulated events which may be directly needed. In our case we used the grep command to filter a trace file, by this command we can create a new file which consists of only those lines from the original file that contain a given character sequences.

The output trace file in our case will contain only the TCP packets which are received by the node 5 (the destination node). The termination of the program is done using a "finish" procedure, which is given below

```
# ending nam and the simulation
proc finish {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
#call xgraph to display the results
    exec xgraph 6ndsdv.tr -t "6 nodes dsdv routing protocol" -x "time" -y
"packets" -geometry 500X400 &
    exec nam -r 5m 6ndsdv.nam &
    exit 0
}
```

The word proc declares a procedure in this case called finish and without arguments. The word global is used to tell that we are using variables declared outside the procedure. The simulator method flush-trace will dump the traces on the respective files. The TCL command "close" closes the trace and NAM files defined before, "exec" command used to execute the NAM program for visualization. The command "exit" will end the application and return the number 0 as status to the system (zero is the default for a clean exit).

At the end of the NS program, we shall call the procedure "finish" and specify at what time the termination should occur, in our case we used the command: $ns at 200.0 "finish", to call the "finish" procedure at time 200 sec. The simulation can then begin using the command: $ns run.

Definition of Links and Nodes

The following commands used to help to configure the nodes in the ad-hoc networks with the parameters above:

```
# configure the nodes
        $ns node-config -adhocRouting $val(rp) \
                   -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 ON

      for {set i 0} {$i < $val(nn) } { incr i } {
            set node_($i) [$ns node]
      }
```

The four last options in the node configure can each be given a value of ON or OFF. The agentTrace will give in our case the trace of TCP, routerTrace provides tracing of packets involved in the routing. macTrace is related to tracing MAC protocol packets, and movementTrace is used to allow tracing the motion of nodes (for NAM).

The initial location of node 0 is given as follows:

```
# Provide initial location of mobile nodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 100.0
$node_(1) set Y_ 150.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 200.0
$node_(2) set Y_ 100.0
$node_(2) set Z_ 0.0
```

```
$node_(3) set X_ 150.0
$node_(3) set Y_ 240.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 400.0
$node_(4) set Y_ 250.0
$node_(4) set Z_ 0.0

$node_(5) set X_ 490.0
$node_(5) set Y_ 285.0
$node_(5) set Z_ 0.0
```

A linear movement of a node is generated by specifying the time in which it starts, the x and y values of the target point and the speed. So the nodes movement will be written as:

```
# Generation of movements
$ns at 10.0 "$node_(0) setdest 250.0 250.0 3.0"
$ns at 15.0 "$node_(5) setdest 45.0 285.0 5.0"
$ns at 110.0 "$node_(0) setdest 480.0 350.0 5.0"
```

Also we need to create the initial node position for NAM using

```
# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}
```

We tell nodes when the simulation ends with

```
# Telling nodes when the simulation ends

for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}
```

Agents and Applications

Having defined the topology (nodes and links) we should now make traffic flow through them. To that end, we need to define routing, agents (protocols) and applications that use them. In our case, we create the TCP connections and the ftp application over it.

```
# Set a TCP connection between node_(0) and node_(5)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
#$tcp set window_ 2000
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(5) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
```

```
$ftp attach-agent $tcp
$ns at 10.0 "$ftp start"
```

TCP is a dynamic reliable congestion control protocol. It uses acknowledgement created by the destination to know whether packets are well received, lost packets are interpreted as congestion signals, that's why TCP requires bidirectional links to return the acknowledgement to the source.

set tcp [new Agent/TCP/Newreno]: this command gives a pointer called "tcp" here to the TCP agent, which is an object in NS. The command $ns attach-agent $node_(0) $tcp, defines the source node of the TCP connection. The command: set sink [new Agent/TCPSink], defines the behavior of the destination node of TCP and assigns to it a pointer called sink. We note that in TCP, the destination node has an active role in the protocol of generation acknowledgements in order to guarantee that all packets arrive at the destination. The command $ns attach-agent $node_(5) $sink, defines the destination node. The command $ns connect $tcp $sink finally makes the TCP connection between the source and destination nodes. So the TCP connection is defined, the FTP application is defined over it.

This is done by using the commands above, note that, both the TCP agent and the FTP application are given pointers: we call them "tcp" and "ftp" respectively. The complete TCL code for 6 nodes DSDV routing protocol is given in table 5.1

```
# 6ndsdv.tcl
# A 6-node example for ad-hoc simulation with DSDV

# 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 val(ifqlen)         50                         ;# max packet in
                                                   ifq
set val(nn)             6                          ;# number of
                                                   mobile nodes
set val(rp)             DSDV                       ;# routing
                                                   protocol
set val(x)              500                        ;# X dimension of
                                                   topography
set val(y)              400                        ;# Y dimension of
                                                   topography
set val(stop)           150                ;# time of simulation end

set ns              [new Simulator]
$ns color 0 red

#open the trace file
set tracefd       [open "| grep \"s\" | grep \"_0_\" | grep \"tcp\" >
tracefile2.tr" w]
$ns trace-all $tracefd

#open the NAM trace file
set namtrace      [open 6ndsdv.nam w]
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

set windowVsTime2 [open 6ndsdv.tr w]

# set up topography object
set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)
# Create nn mobile nodes [$val(nn)] and attach them to the channel.


# configure the nodes
        $ns node-config -adhocRouting $val(rp) \
                   -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 ON

      for {set i 0} {$i < $val(nn) } { incr i } {
            set node_($i) [$ns node]
      }

# Provide initial location of mobile nodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 100.0
```

```
$node_(1) set Y_ 150.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 200.0
$node_(2) set Y_ 100.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 150.0
$node_(3) set Y_ 240.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 400.0
$node_(4) set Y_ 250.0
$node_(4) set Z_ 0.0

$node_(5) set X_ 490.0
$node_(5) set Y_ 285.0
$node_(5) set Z_ 0.0

$ns at 0.0 "$node_(0) add-mark m1 blue circle"
$ns at 0.1 "$node_(5) add-mark m1 green circle"

# Generation of movements
$ns at 10.0 "$node_(0) setdest 250.0 250.0 3.0"
$ns at 15.0 "$node_(5) setdest 45.0 285.0 5.0"
$ns at 110.0 "$node_(0) setdest 480.0 350.0 5.0"

$ns at 0.0 "$node_(0) label \"SOURCE\""
$ns at 0.0 "$node_(5) label \"DESTINATION\""



# Set a TCP connection between node_(0) and node_(5)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
#$tcp set window_ 2000
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(5) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 "$ftp start"

# Printing the window size
proc plotWindow {tcpSource file} {
global ns
set time 0.01
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file" }
$ns at 10.1 "plotWindow $tcp $windowVsTime2"

# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
proc finish {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
#call xgraph to display the results
    exec xgraph 6ndsdv.tr -t "6 nodes dsdv routing protocol" -x "time" -y
"packets" -geometry 500X400 &
    exec nam -r 5m 6ndsdv.nam &
    exit 0
}
$ns at 200.0 "finish"

$ns run
```

5.3.2. Analysis of Simulation Results

When performing the simulation, we observe five phases of operation. In the first, the nodes are too far away and there is no connectivity between the source and the destination. During phase 2 the connection between nodes 0 and 5 use nodes 2, and 4 as a relay and the packets drop will start, whereas in the 3rd phase, there is a direct path between node 0 and 5, in the 4th phase the source and the destination nodes will use the node 3 as a rely, in the last phase the nodes 1 and 5 will use node 2 as a rely.

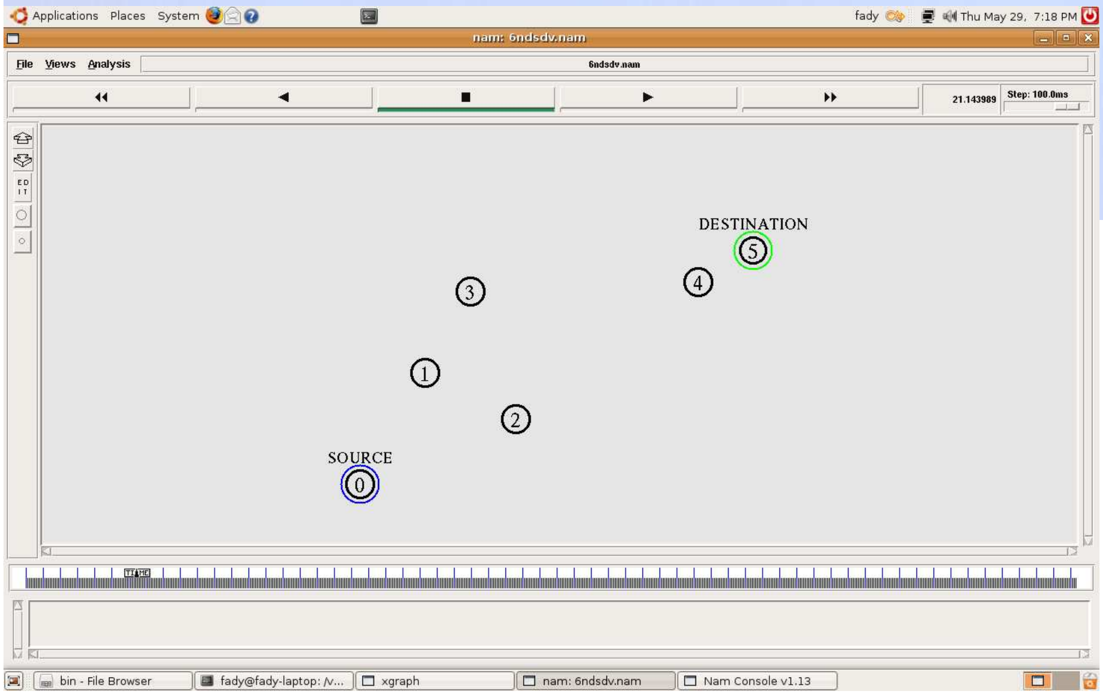


Figure 5.2 phase- 1 for DSDV routing protocol

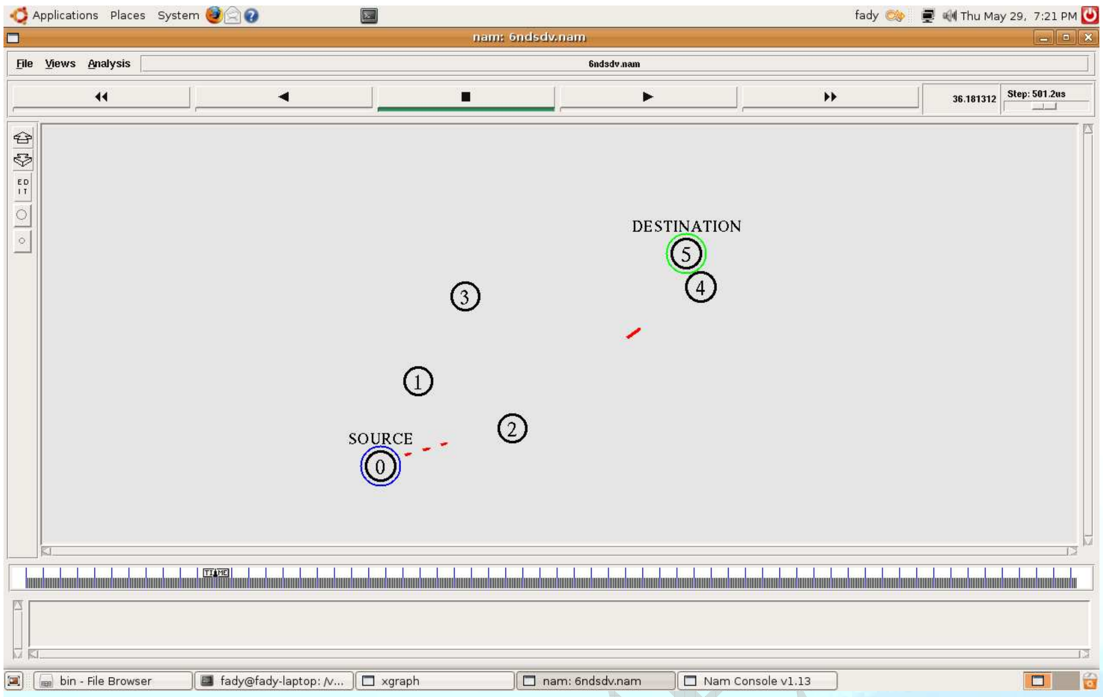


Figure 5.3 phase- 2 for DSDV routing protocol

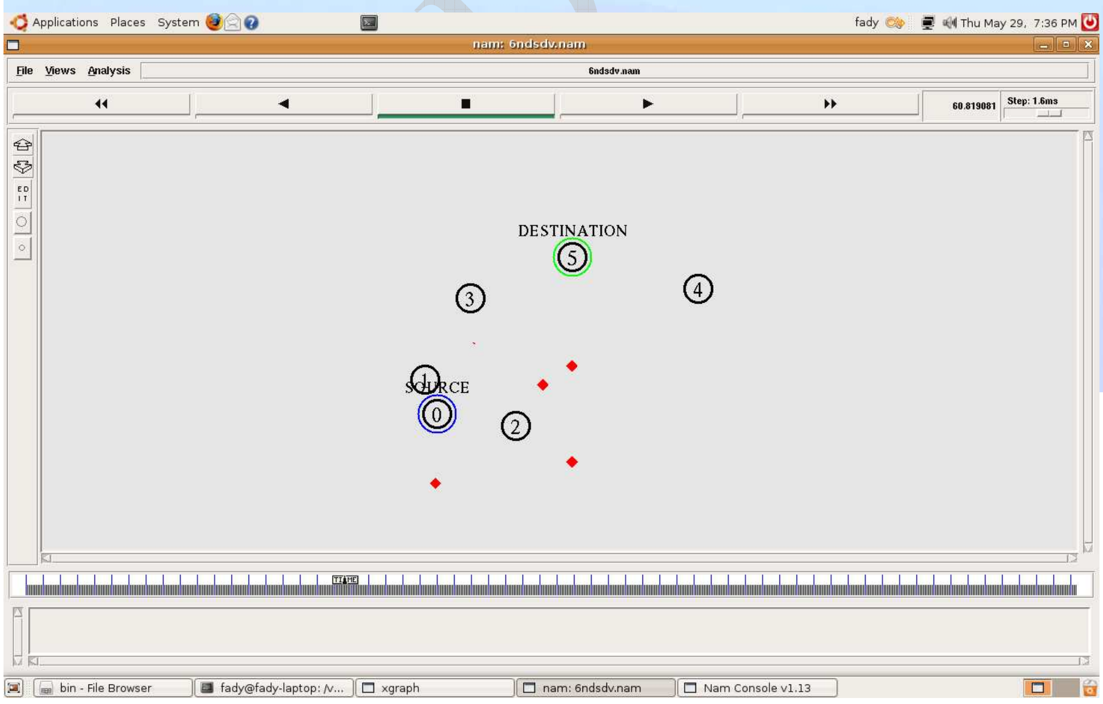


Figure 5.4 packets dropped in case of DSDV routing protocol

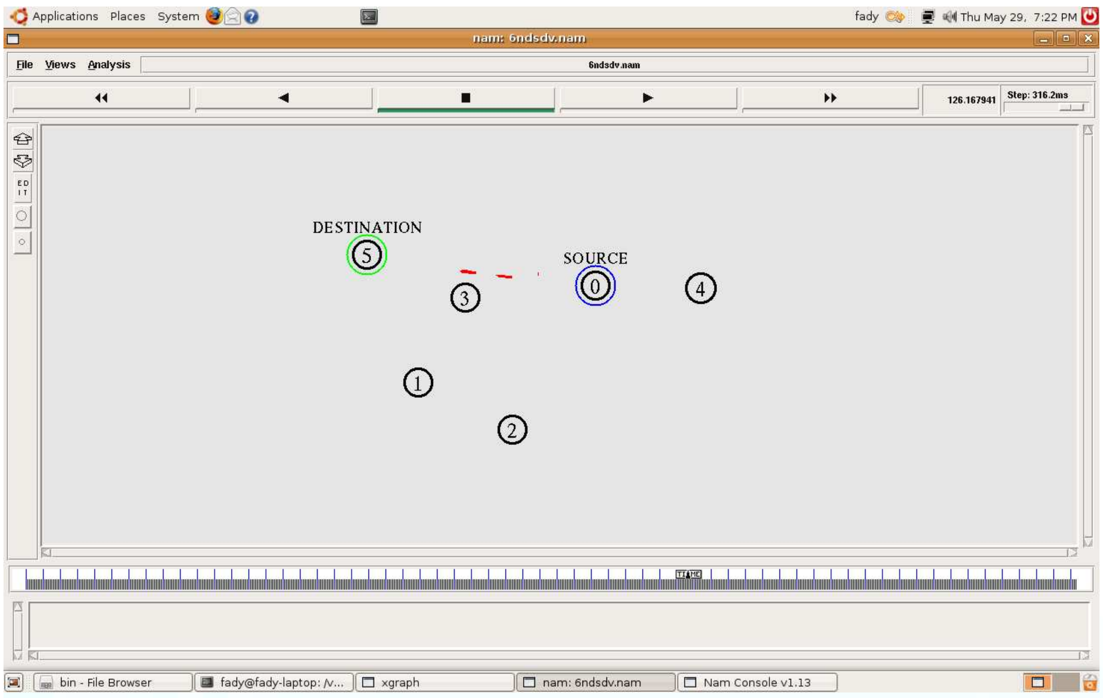


Figure 5.5 phase- 3 for DSDV routing protocol

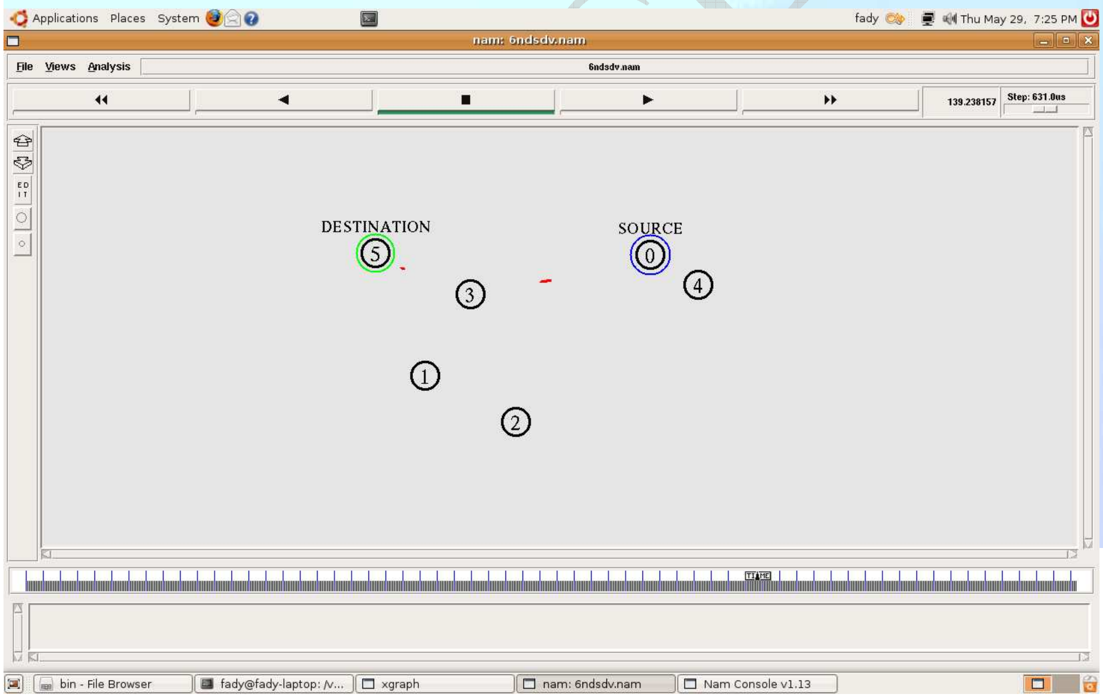


Figure 5.6 phase- 4 for DSDV routing protocol

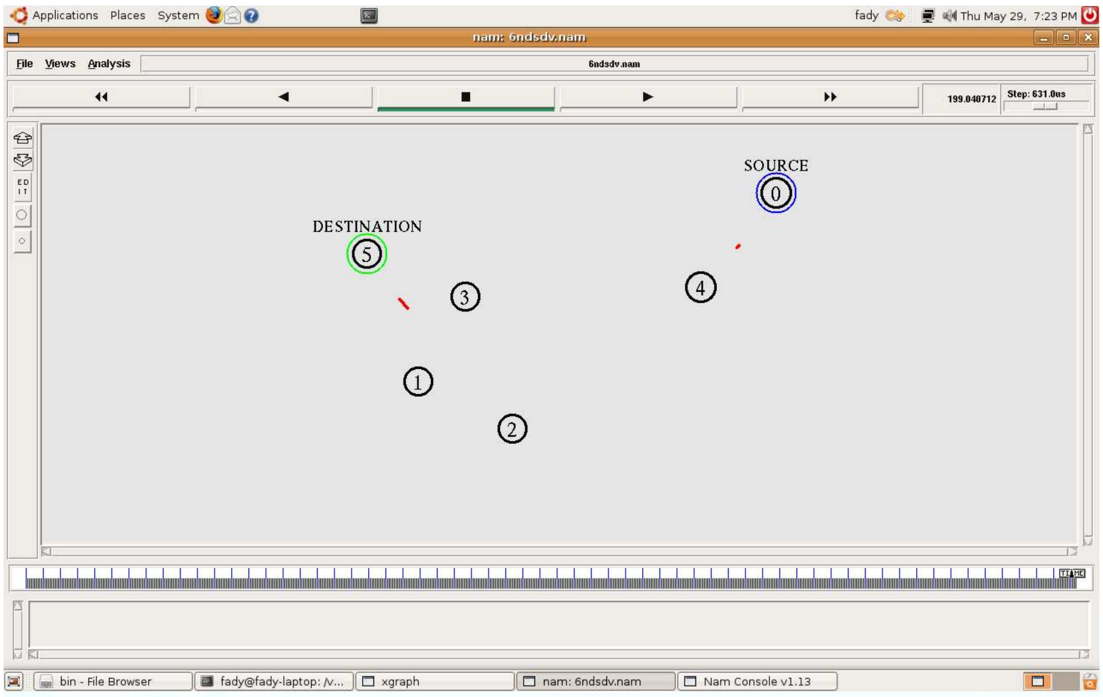


Figure 5.7 phase- 5 for DSDV routing protocol

At the beginning the nodes are too far away and a connection cannot be set. The first TCP signaling packet is transmitted at time 10 but the connection cannot be opened. After 20 seconds (time out) the connection between node 0 and node 5 will start and the packets will start passing between them. After 40 seconds node 0 will be closer to node 5, so the number of packets are increased between them. After 60 seconds, nodes 0 and 5 will be closer to each other so there is a direct connection established between them and the maximum value for packets received will be in this period (between 60-to-120 second). After that the distance between the source and the destination will increase and the mobiles get further apart till the direct link brakes and the packets drop will happen. The routing protocol is too slow to react and to create an alternative route. The window size evolution is given in figures 5.8 and 5.9 below.

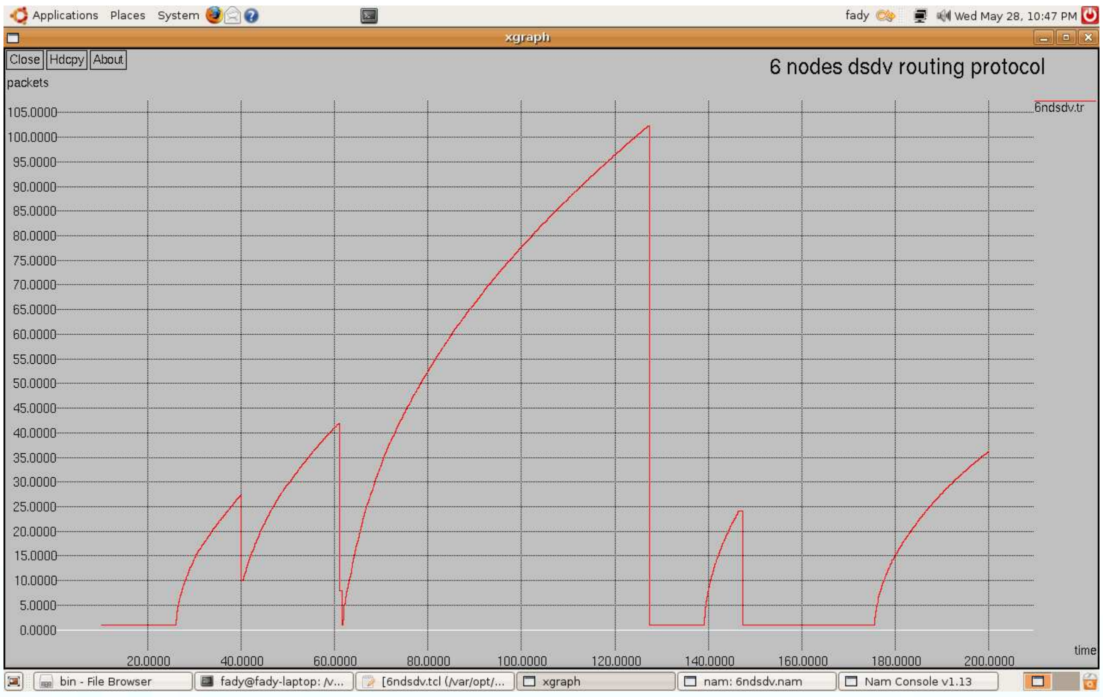


Figure 5.8 TCP window size in a six node scenario with DSDV routing protocol

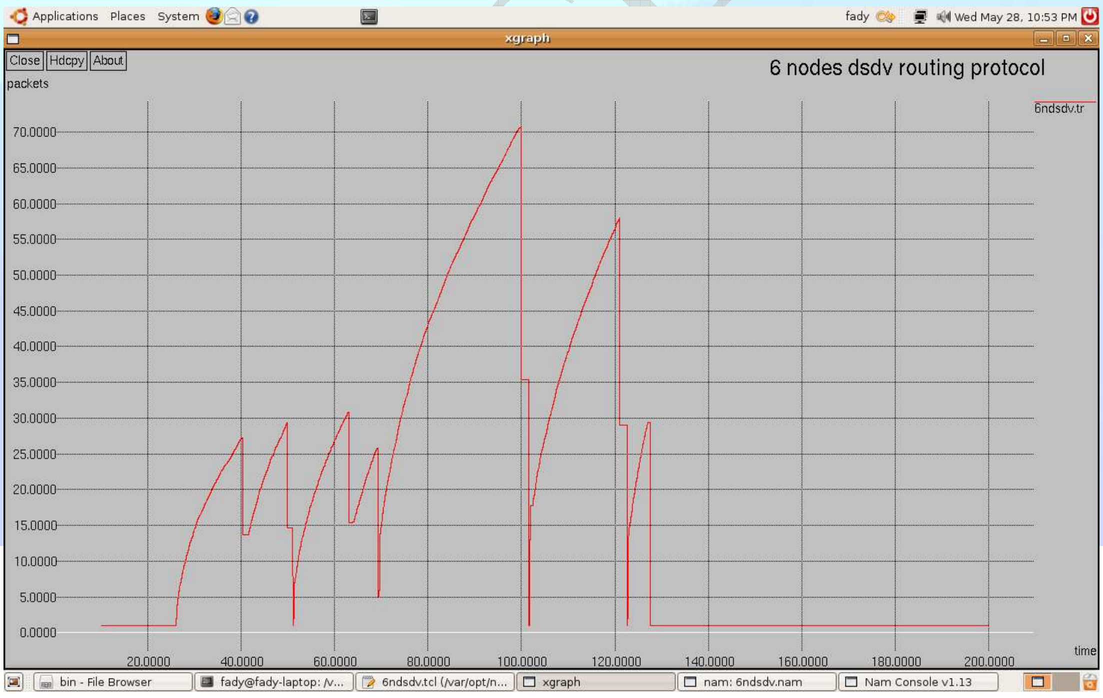


Figure 5.9 maximum windows of 2000 size evolutions for standard TCP

5.4 Simulation Scenario for AODV Routing Protocol

We start by presenting simple script that runs a single TCP connection over 6-nodes network over an area of a size of 500m over 400m depicted in fig 5.10.

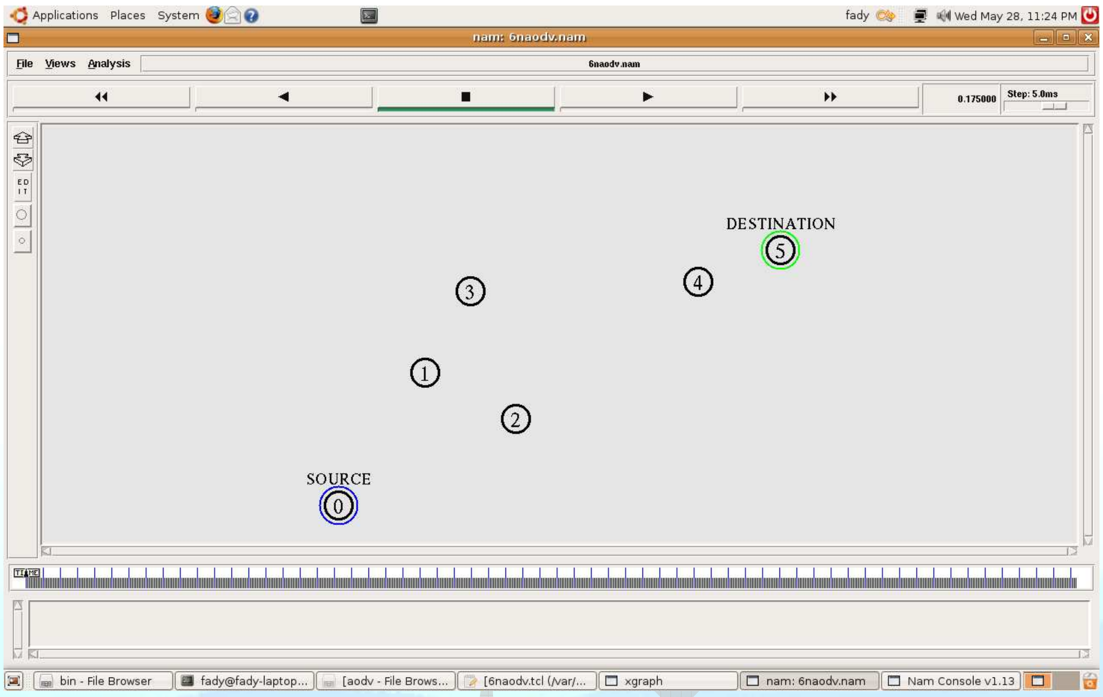


Figure 5.10 topology of a six nodes ad-hoc network

The initial location of nodes 0, 1, 2, 3, 4, 5 are respectively (5, 5), (100,150), (200,100), (150,240), (400,250), (490,285), the Z coordinate is assumed throughout to be 0.

The simulation lasts at 200 second, $ns at 200.0 "finish". At time 10, a TCP connection is initiated between node 0 and node 5, with using AODV routing protocol.

5.4.1. Complete TCL script

The complete TCL code for 6 nodes AODV routing protocol is given in table 5.2

```
# 6naodv.tcl
# A 6-nodes network for ad-hoc simulation with AODV
# 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 val(ifqlen)         50                         ;# max packet in
                                                   ifq
set val(nn)             6                          ;# number of
                                                   mobile nodes
set val(rp)             AODV                       ;# routing
                                                   protocol
set val(x)              500                        ;# X dimension of
                                                   topography
set val(y)              400                        ;# Y dimension of
                                                   topography
set val(stop)           150               ;# time of simulation end

set ns              [new Simulator]
$ns color 0 red

#open the trace file
set tracefd       [open "| grep \"r\" | grep \"_5_\" | grep \"tcp\" >
tracefile2.tr" w]
$ns trace-all $tracefd

#open the NAM trace file
set namtrace      [open 6naodv.nam w]
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

set windowVsTime2 [open 6naodv.tr w]

# set up topography object
set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)

#

# Create nn mobile nodes [$val(nn)] and attach them to the channel.
#

# configure the nodes
        $ns node-config -adhocRouting $val(rp) \
                   -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 ON

      for {set i 0} {$i < $val(nn) } { incr i } {
            set node_($i) [$ns node]
      }

# Provide initial location of mobilenodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 100.0
$node_(1) set Y_ 150.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 200.0
$node_(2) set Y_ 100.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 150.0
$node_(3) set Y_ 240.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 400.0
$node_(4) set Y_ 250.0
$node_(4) set Z_ 0.0

$node_(5) set X_ 490.0
$node_(5) set Y_ 285.0
$node_(5) set Z_ 0.0

$ns at 0.0 "$node_(0) add-mark m1 blue circle"
$ns at 0.1 "$node_(5) add-mark m1 green circle"

# Generation of movements
$ns at 10.0 "$node_(0) setdest 250.0 250.0 3.0"
$ns at 15.0 "$node_(5) setdest 45.0 285.0 5.0"
$ns at 110.0 "$node_(0) setdest 480.0 350.0 5.0"

$ns at 0.0 "$node_(0) label \"SOURCE\""
$ns at 0.0 "$node_(5) label \"DESTINATION\""


# Set a TCP connection between node_(0) and node_(5)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
#$tcp set window_ 2000
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(5) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 "$ftp start"

# Printing the window size
proc plotWindow {tcpSource file} {
```

```
global ns
set time 0.01
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file" }
$ns at 10.1 "plotWindow $tcp $windowVsTime2"

# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam

$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
proc finish {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
#call xgraph to display the results
    exec xgraph 6naodv.tr -t "6 nodes aodv routing protocol" -x "time" -y
"packets" -geometry 500X400 &
    exec nam -r 5m 6naodv.nam &
    exit 0
}
$ns at 200.0 "finish"

$ns run
```

5.4.2 TCP over AODV

The simulation with the same parameters as before is repeated with AODV. When performing the simulation, we observe five phases of operation. In the first, the nodes are too far away and there is no connectivity between the source and the destination. During phase 2 the connection between nodes 0 and 5 use nodes 2, and 4 as a relay and the packets drop will start, whereas in the 3rd phase, there is a direct path between node 0 and 5, in the case of AODV routing protocol the 4th is not there and the connection between the source and the destination will continue up to the last phase, and this case helped to increase the number of the packets received , in the last phase the nodes 1 and 5 will use node 2 as a rely.

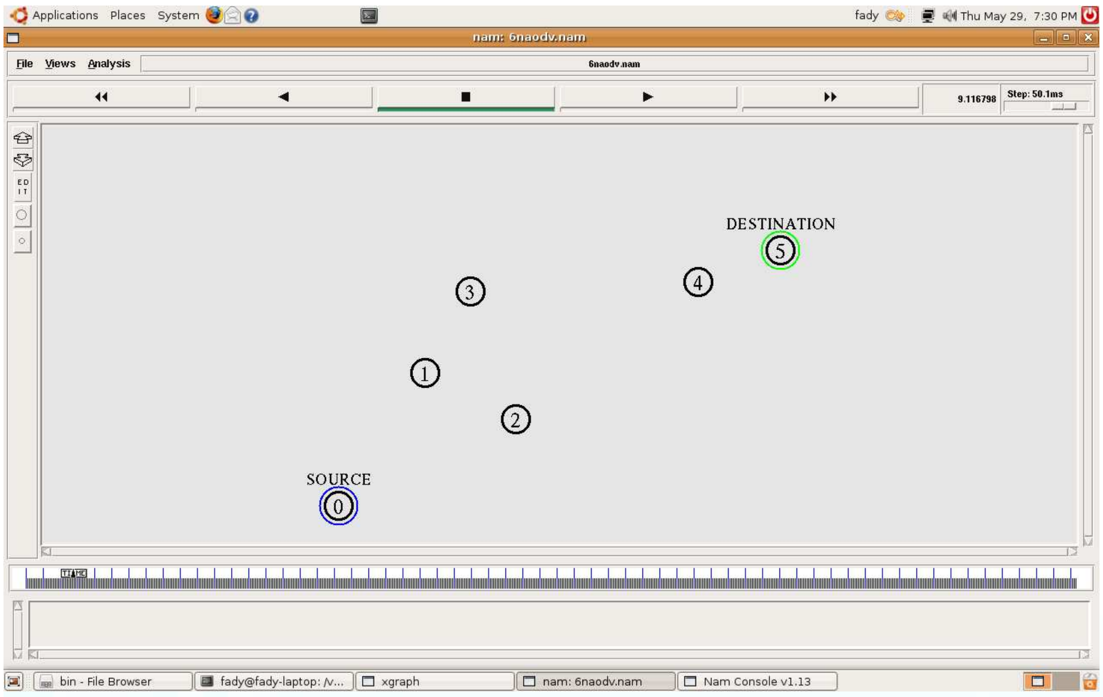


Figure 5.11 phase- 1 for AODV routing protocol

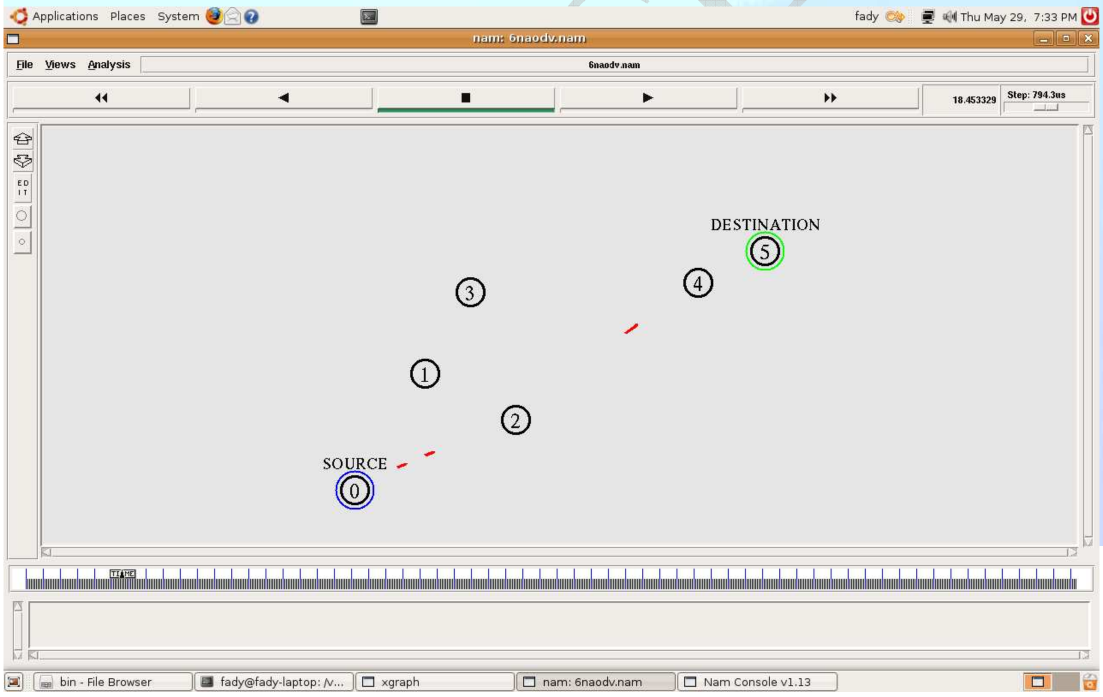


Figure 5.12 phase- 2 for AODV routing protocol

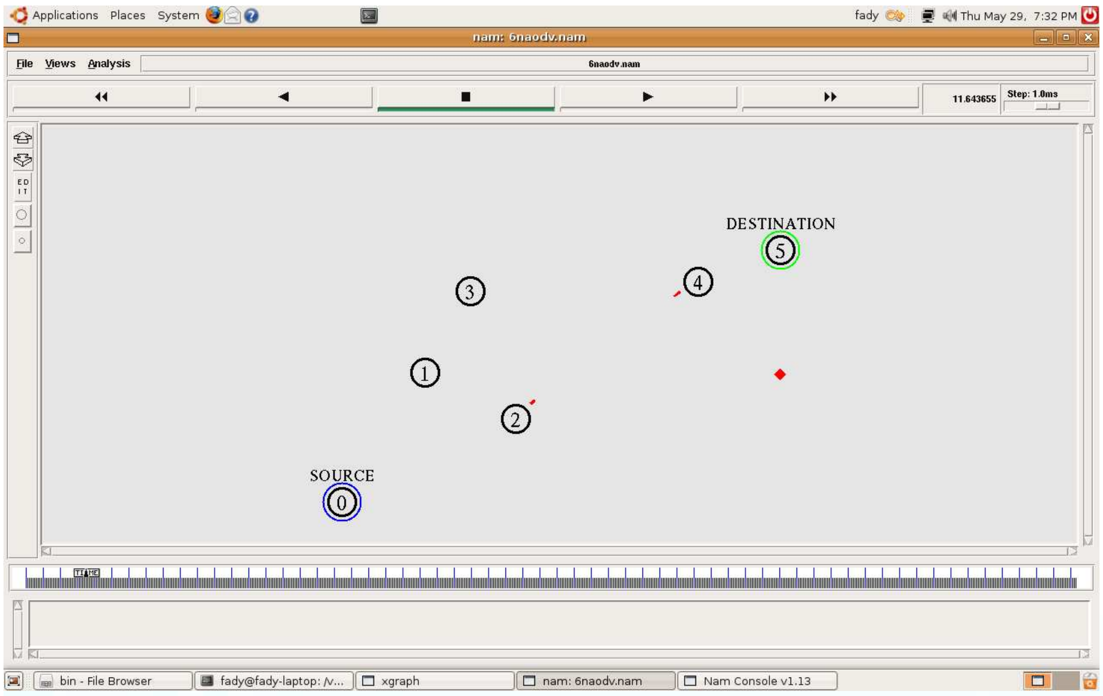


Figure 5.13 the packet dropped in the case of AODV routing protocol

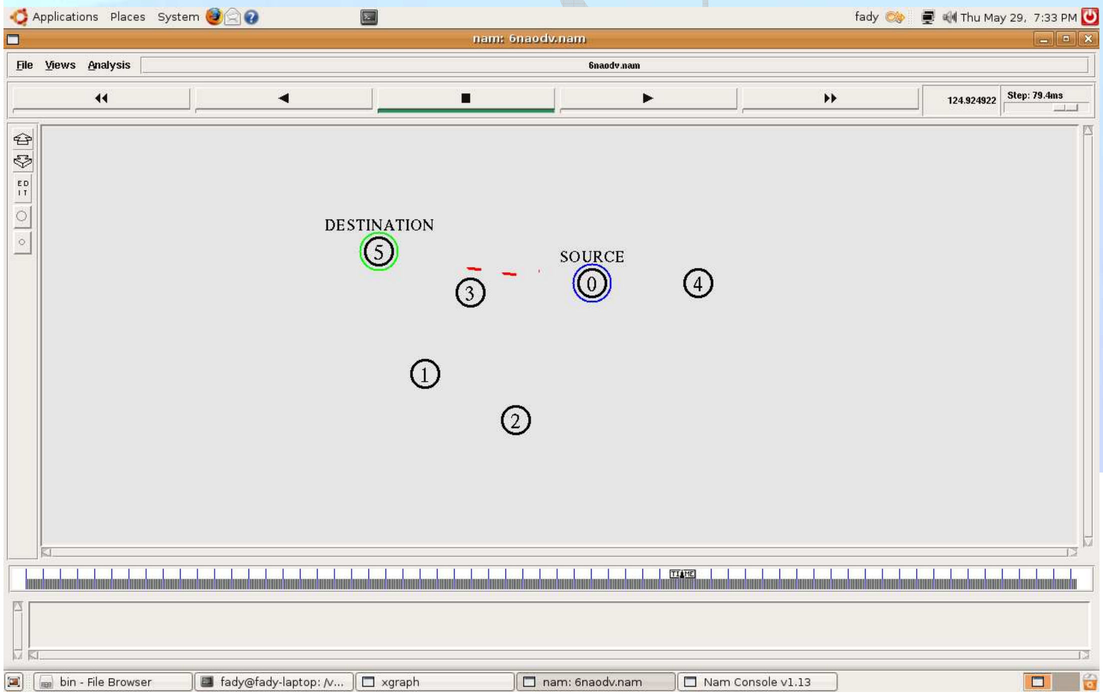


Figure 5.14 phase- 3 for AODV routing protocol

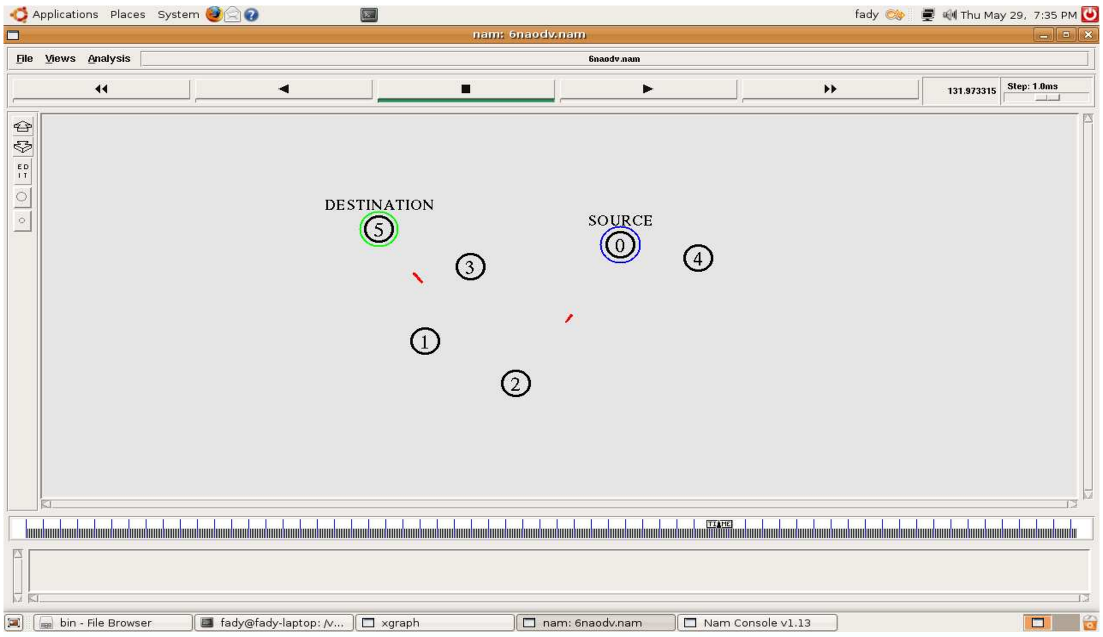


Figure 5.15 phase- 5 for AODV routing protocol

The window size is given in figure 5.16. It had throughout a long single phase in which the same five phases was used, in which nodes 2, and 4 relayed the packets.

Due to the fact that changes in paths were avoided, there were no losses so the window will start high. And also the round trip time (needed to increase the window by one unit) is longer since a direct path is not used here. We thus see that finding a shorter path results in a better TCP performance.

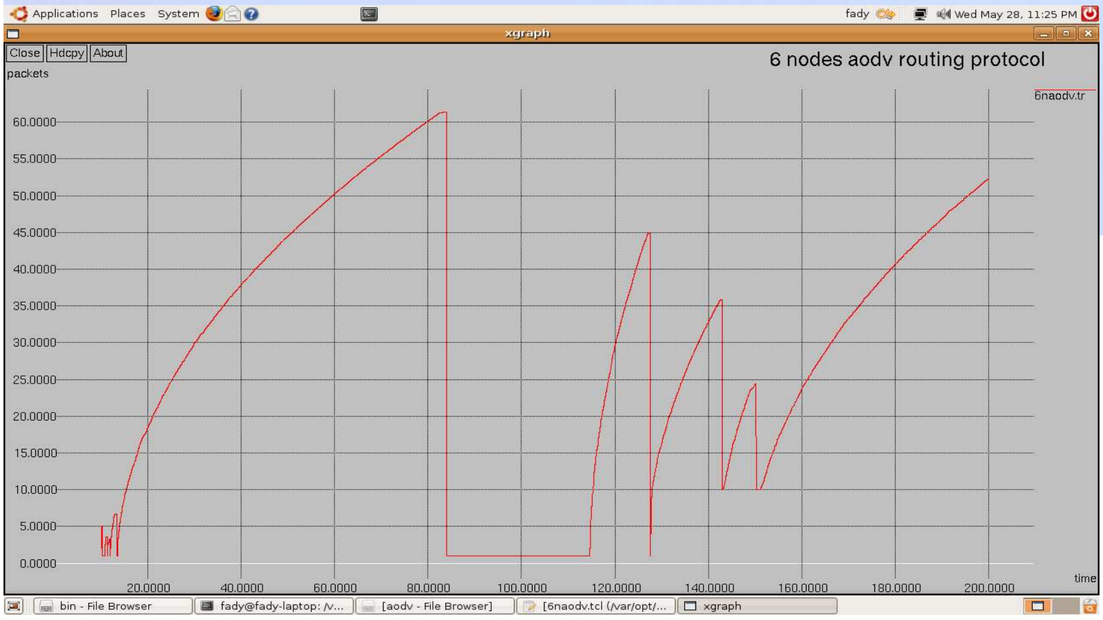


Figure 5.16 TCP window size in a six node scenario with AODV routing protocol

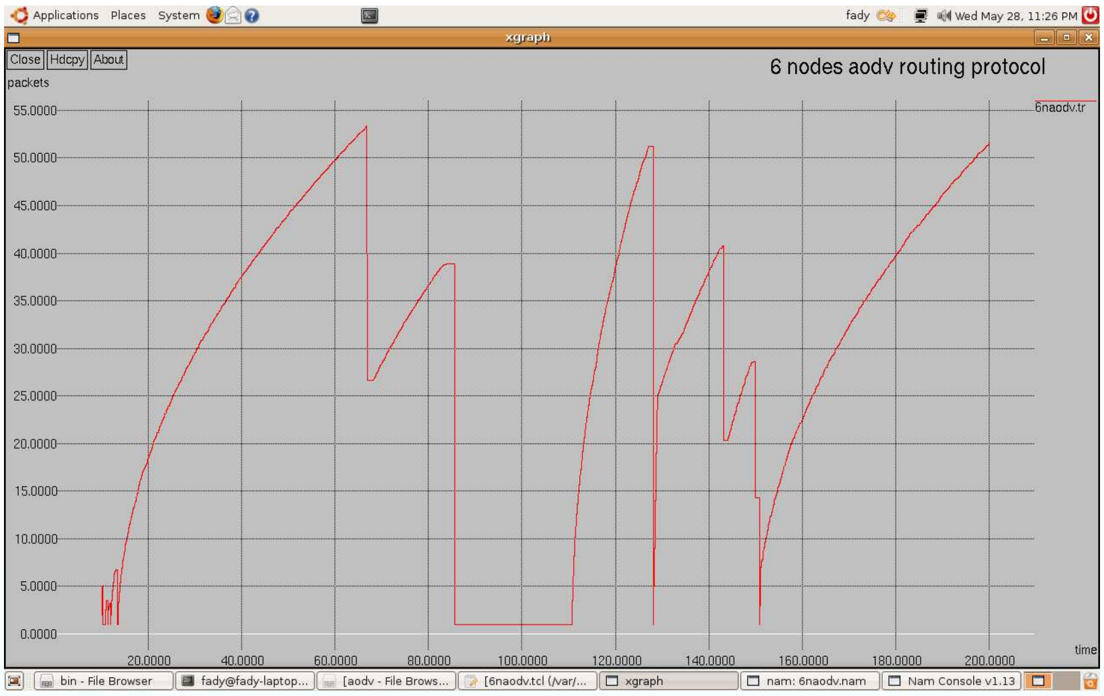


Figure 5.17 maximum windows of 2000 size evolutions for standard TCP

In the two routing protocols that we considered, losses occurred either when the geographical range was too far for reception or when there was a route change, and there were no losses due to buffer overflow.

This is due to the fact that we used the default value of the maximum window size of TCP. So using maximum window size of TCP implies losses due to overflow.

5.5 The Interaction of TCP with MAC Protocol

In the previous section we considered a small number of mobiles, and saw how mobility phenomena influenced the performance of TCP. When there are a large number of terminals, new particular phenomena due to the MAC and physical layers may have a critical influence on TCP performance.

Each transmission of a data packet at the MAC layer is part of a four-way handshake protocol. The mobile that wishes to send a packet, which we call M1, first sends an RTS (request to Send) packet. If the destination mobile, which we call M2, can receive the packet, it sends a

CTS (Clear to Send) packet. If M1 receives the CTS it can then send the DATA packet (e.g. TCP data or ACK packet). Finally, M2 sends a (MAC layer) ACK so that M1 knows that the data packet has been well received.

This handshake protocol is intended to reduce the collision probability. Collisions may occur since a mobile, say M3, may wish to send a packet to M2 at the same time as M1 does, M3 may be out of range to sense the transmission from M1, so a collision of M1's and M3's packets may occur at M2, This phenomenon the "hidden terminal phenomenon". With the handshake protocol, M3 will not attempt to send any packet when it hears the CTS packet sent by M2 to M1.

If a sender M1 does not receives CTS packet then it differs its transmission and makes later attempts of sending an RTS, A sender drops the DATA packet if it has resent the RTS message seven times and has not heard a CTS reply from the receiver. A data packet is also dropped after four retransmission without receiving a (MAC layer) ACK.

5.5.1 The Simulated Scenario

We use the standard two-ray ground propagation model. The IEEE802.11 MAC and an omni directional antenna model for ns. We use the AODV routing algorithm; at each node we tested the NewReno version of TCP, which is the most deployed one. in this scenario we used 9 nodes required 150 sec for simulation. The script for this case is given in table 5.3 below, when the configuring the nodes we shall use the option "macTrace ON" in order to have detailed tracing of MAC protocols packets, this will allow us to analyze the reason of each TCP packet loss that occurs.   The complete TCL code for 9 nodes AODV routing protocol is given in table 5.3

```
# A 9-node example for ad-hoc simulation with AODV

# 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 val(ifqlen)         50                 ;# max packet in ifq
set val(nn)             9                           ;# number of
                                                    mobilenodes
set val(rp)             AODV                        ;# routing
                                                    protocol
set val(x)              2200                        ;# X dimension of
                                                    topography
set val(y)              500                         ;# Y dimension of
                                                    topography
set val(stop)           150                 ;# time of simulation end

set ns              [new Simulator]

set tracefd       [open aodv.tr w]
$ns trace-all $tracefd

set namtrace      [open aodv.nam w]
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

set windowVsTime2 [open winaodv.tr w]
# set up topography object
set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)


#
#  Create nn mobilenodes [$val(nn)] and attach them to the channel.
#

# configure the nodes
        $ns node-config -adhocRouting $val(rp) \
                   -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 ON \
                   -movementTrace ON \



      for {set i 0} {$i < $val(nn) } { incr i } {
            set node_($i) [$ns node]
      }

# Provide initial location of mobilenodes
```

```
for {set i 0 } {$i <  $val(nn)} { incr i } {
$node_($i) set X_ [expr ($i+1)*200.0]
$node_($i) set y_ 250.0
$node_($i) set z_ 0.0
}
# Generation of movements
$ns at 10.0 "$node_(1) setdest 400.0 300.0 3.0"
$ns at 20.0 "$node_(3) setdest 800.0 300.0 3.0"
$ns at 30.0 "$node_(5) setdest 1200.0 300.0 3.0"
$ns at 40.0 "$node_(7) setdest 1600.0 300.0 3.0"

# Set a TCP connection between node_(0) and node_(1)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(1) $sink
$ns connect $tcp $sink
# Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
# Scheduling events
$ns at 10.0 "$ftp start"



# Set a TCP connection between node_(1) and node_(2)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(1) $tcp
$ns attach-agent $node_(2) $sink
$ns connect $tcp $sink
# Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
# Scheduling events
$ns at 10.0 "$ftp start"


# Set a TCP connection between node_(2) and node_(3)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(2) $tcp
$ns attach-agent $node_(3) $sink
$ns connect $tcp $sink
# Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
# Scheduling events
$ns at 20.0 "$ftp start"


# Set a TCP connection between node_(3) and node_(4)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(3) $tcp
$ns attach-agent $node_(4) $sink
$ns connect $tcp $sink
# Setup a FTP over TCP connection
set ftp [new Application/FTP]
```

```
$ftp attach-agent $tcp
# Scheduling events
$ns at 20.0 "$ftp start"

# Set a TCP connection between node_(4) and node_(5)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(4) $tcp
$ns attach-agent $node_(5) $sink
$ns connect $tcp $sink
# Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
# Scheduling events
$ns at 30.0 "$ftp start"

# Set a TCP connection between node_(5) and node_(6)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(5) $tcp
$ns attach-agent $node_(6) $sink
$ns connect $tcp $sink
# Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
# Scheduling events
$ns at 30.0 "$ftp start"


# Set a TCP connection between node_(6) and node_(7)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(6) $tcp
$ns attach-agent $node_(7) $sink
$ns connect $tcp $sink
# Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
# Scheduling events
$ns at 40.0 "$ftp start"


# Set a TCP connection between node_(7) and node_(8)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(7) $tcp
$ns attach-agent $node_(8) $sink
$ns connect $tcp $sink
# Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
# Scheduling events
$ns at 40.0 "$ftp start"


# Printing the window size
proc plotWindow {tcpSource file} {
global ns
set time 0.01
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file" }
$ns at 12.1 "plotWindow $tcp $windowVsTime2"


# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {

# 50 defines the node size for nam
$ns initial_node_pos $node_($i) 50
}


# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
proc finish {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
#call xgraph to display the results
    exec xgraph winaodv.tr -t "9 nodes aodv routing protocol" -x "time" -y
"packets" -geometry 2200X500 &
    exec nam -r 5m aodv.nam &
    exit 0
}
$ns at 150.0 "finish"


$ns run
```

5.5.2 Simulation Results Our simulation results for n = 9 are summarized in the figures

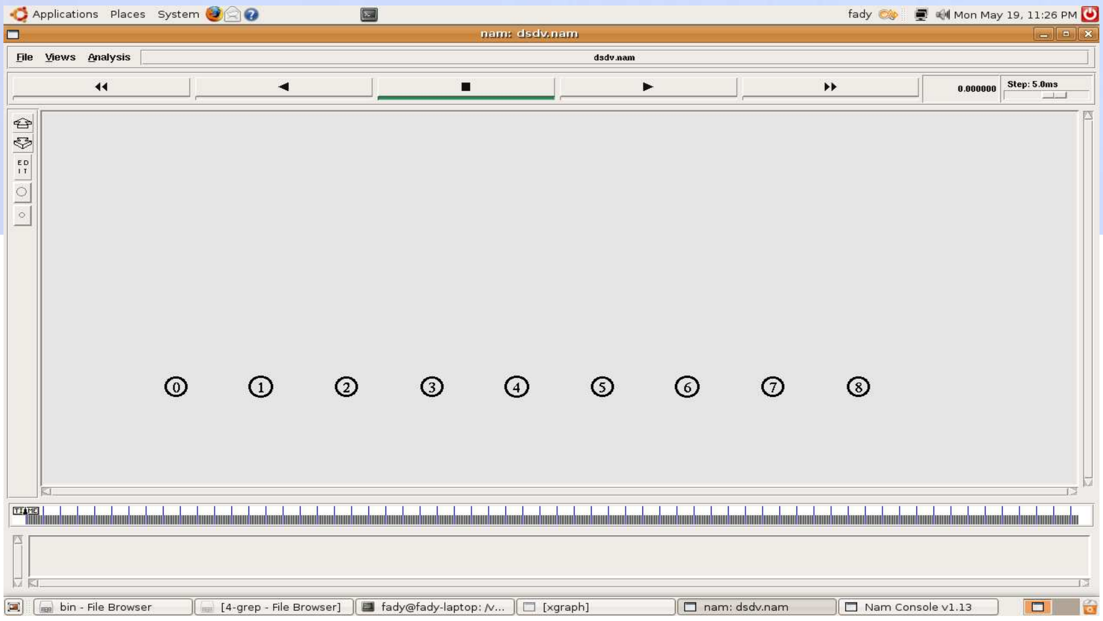


Figure 5.18 topology of a nine nodes ad-hoc network

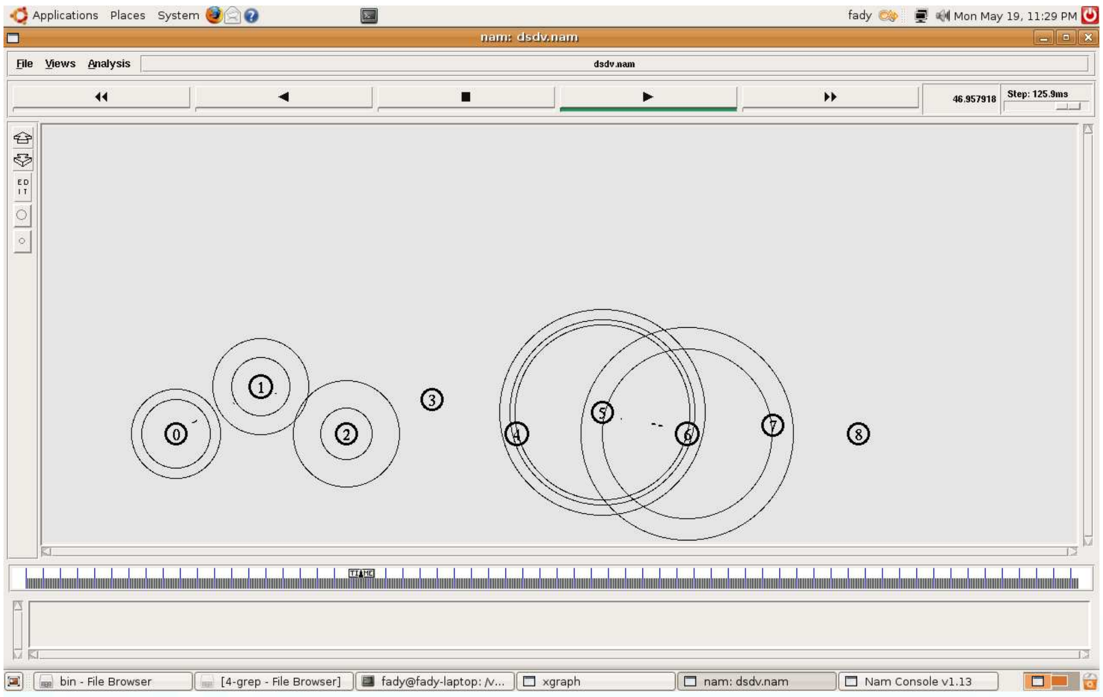


Figure 5.19 a nine nodes ad-hoc network with packets passing over TCP connections

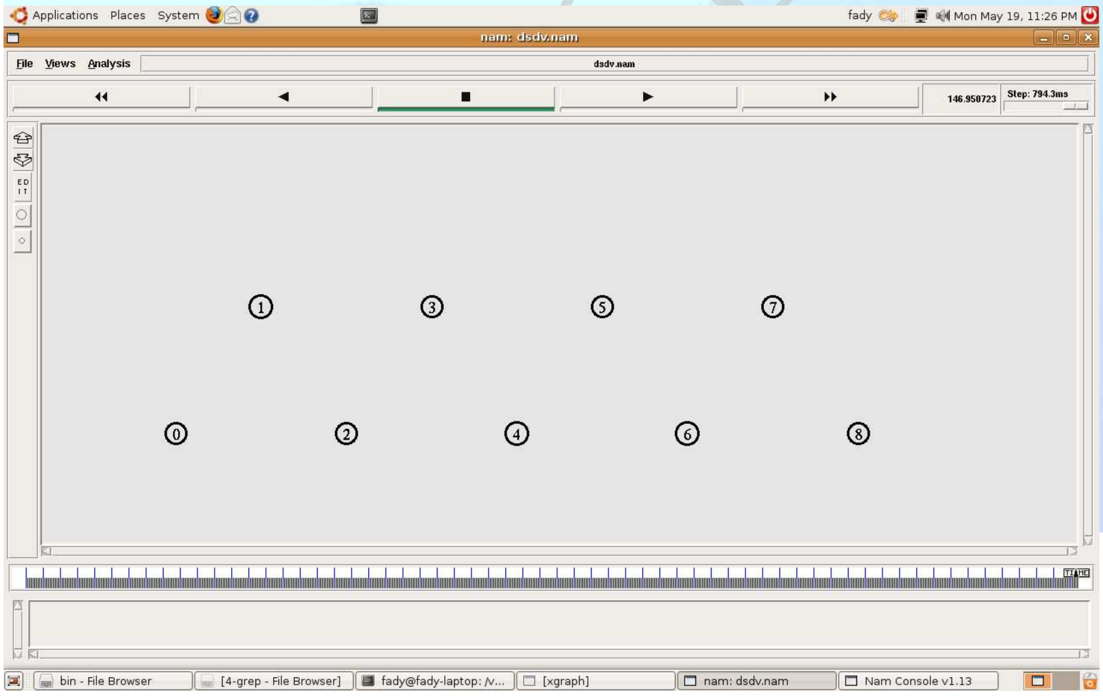
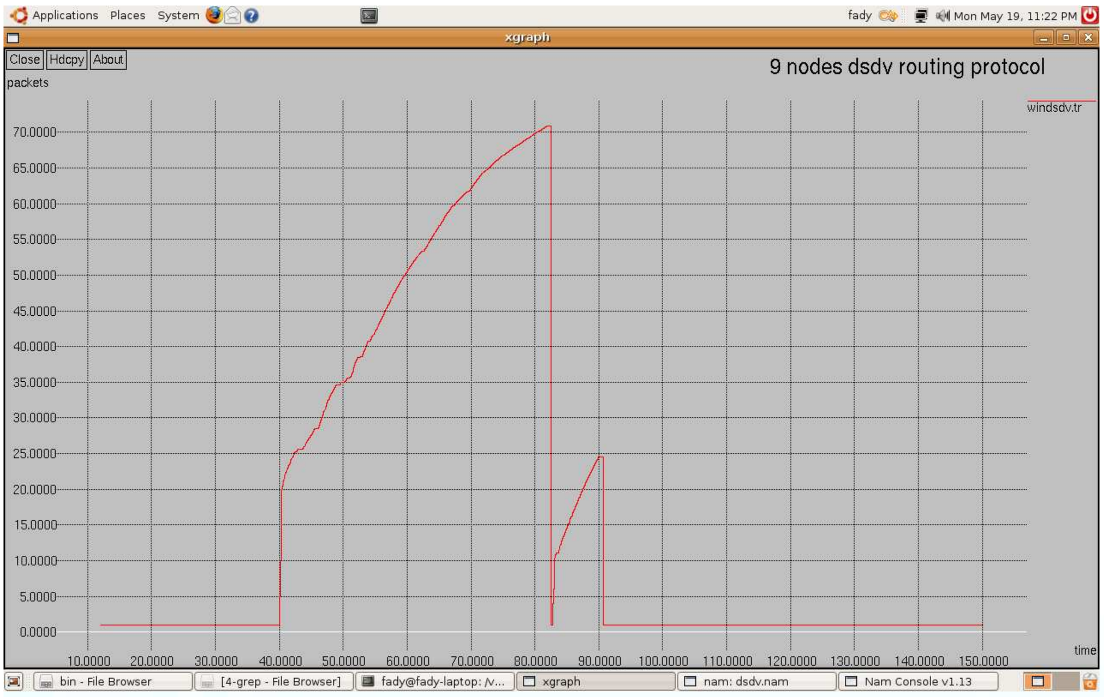
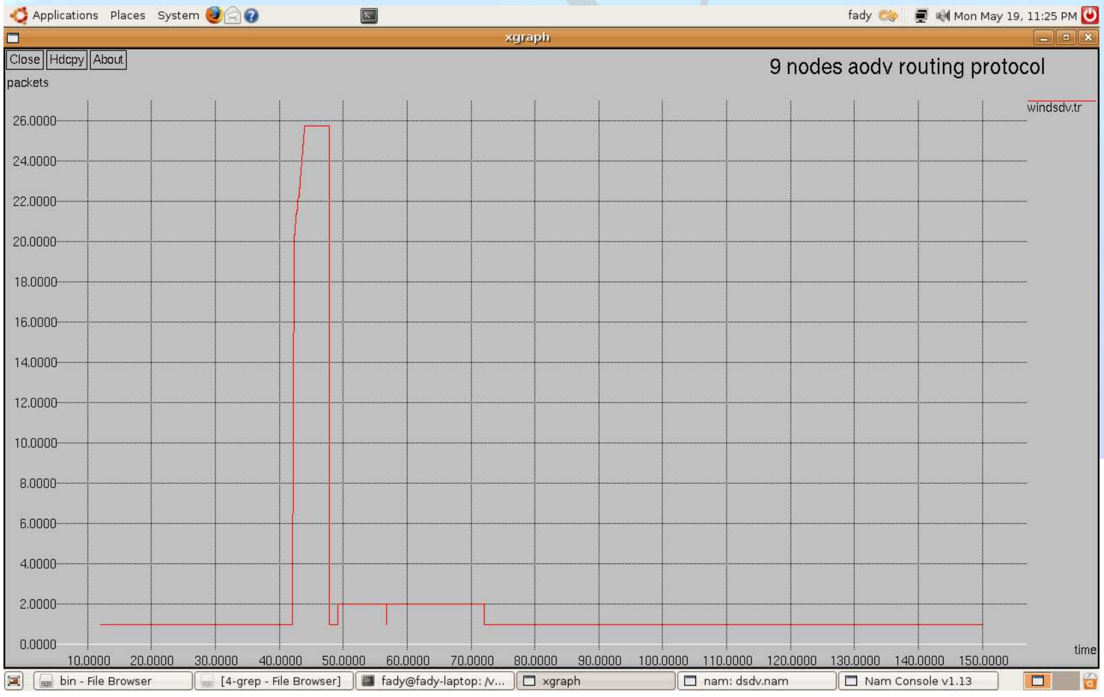


Figure 5.20 the end of the simulation for nine nodes ad-hoc network



<u>Figure 5.21 TCP window size in a nine nodes scenario with DSDV routing protocol</u>



<u>Figure 5.22 TCP window size in a nine nodes scenario with AODV routing protocol</u>

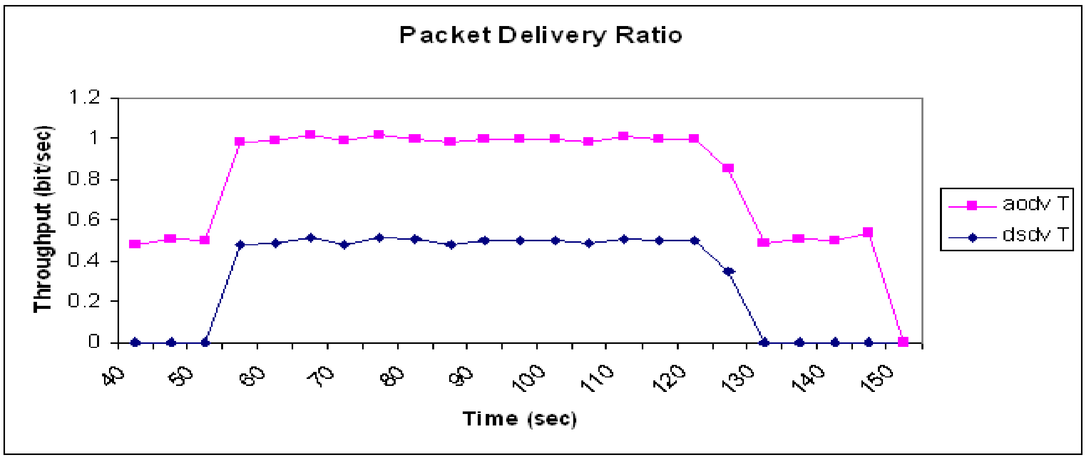
5.6 Simulation Results for the Performance

The performance measures which have been used for evaluating the performance of the two routing protocols by using the Trace file and compare the results with different nodes number.

5.6.1 Packet Delivery Ratio

It is also called "Throughput", it is the rate at which a network sends or receives data. It is a good channel capacity of network connections and rated in terms bits per second (bit/sec). Throughput (T p) = Pa / Pf, where Pa is the packets received and Pf is the amount of packets sent over certain time interval.

In our work, we have done the throughput evaluation through two different ad-hoc wireless networks just to confirm our results. The first ad-hoc wireless network contains 3-nodes only (see the diagrams below)



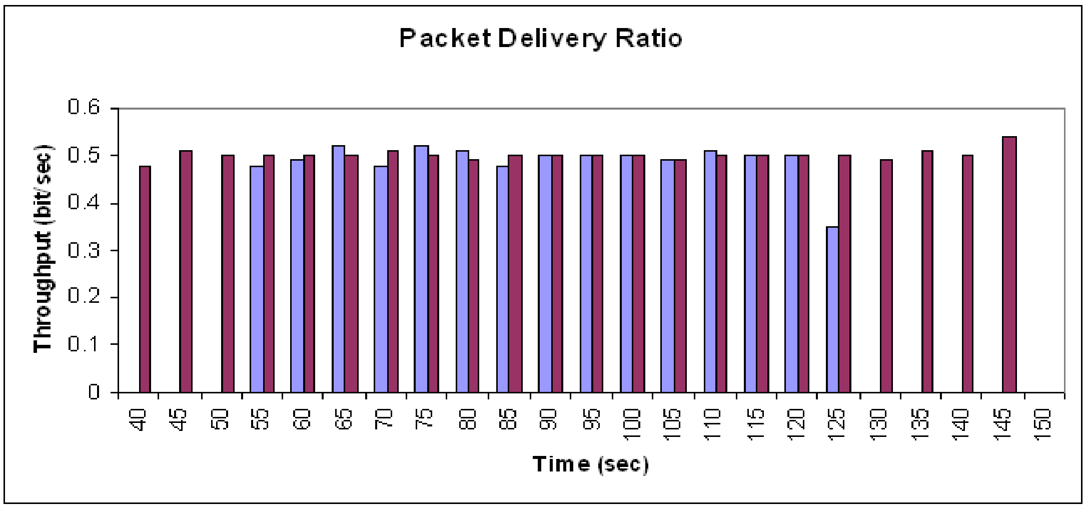
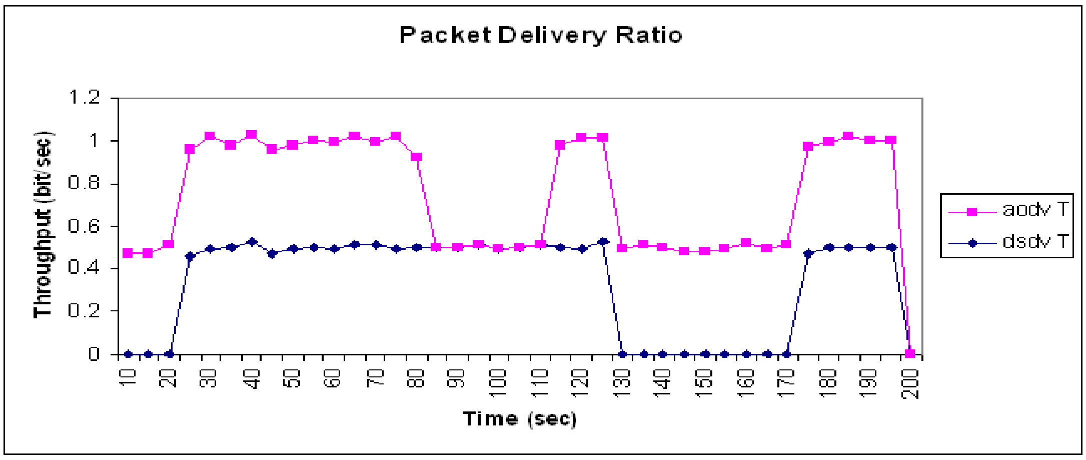


Figure 5.23 a three nodes Packet Delivery Ratio

In the both cases above show that, the number of packets received was higher in the case of AODV routing protocol than DSDV routing protocol, because in the case of AODV routing protocol, the packets start receiving by the node 5 from 40 seconds up to 150 seconds, but in the case of DSDV the receiving will start from 55 seconds up to 125 second only.

The second ad-hoc network contains 6-nodes only (see the diagrams below), also we got the same results, the number of packets received by the node 5 was higher in the case of AODV routing protocol than DSDV routing protocol, because in the case of AODV routing protocol, the packets start receiving by the node 5 from 10 seconds, but in the case of DSDV the receiving will start from 30 seconds, and also if we compare the period between 135 seconds to 175 seconds which is contain the higher level for receiving in the case of AODV routing protocol and lower level for receiving in the case of DSDV routing protocol.



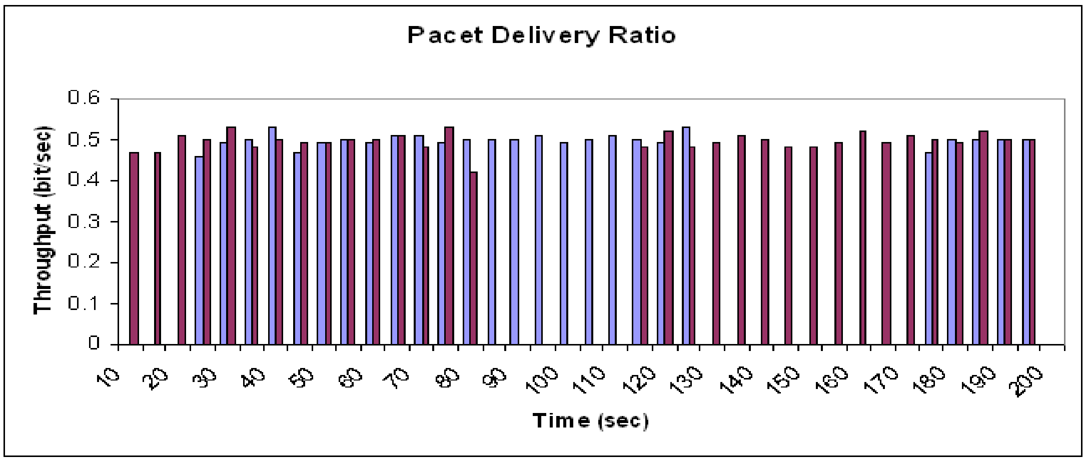
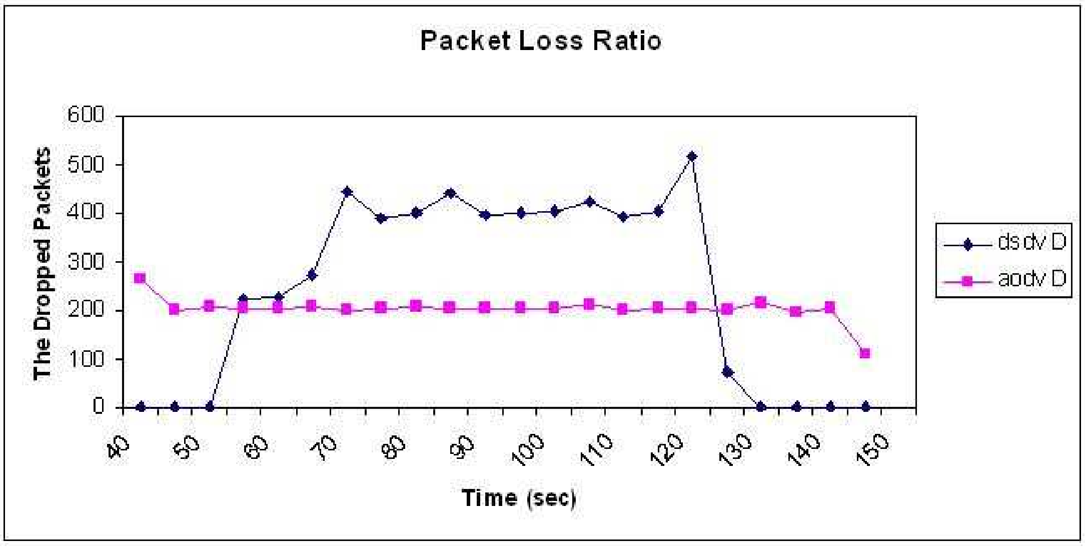


Figure 5.24 a six nodes Packet Delivery Ratio

5.6.2 Packet Loss Ratio

Packet loss is where network traffic fails to reach its destination in a timely manner. Most commonly packets get dropped before the destination can be reached. Packet dropped/loss (Pd) = Ps - Pa, where Ps is the amount of packet sent and Pa amount of packet received.

In our work, we have done the Packet Loss Ratio evaluation through two different ad-hoc wireless networks just to confirm our results. The first ad-hoc network contains 3-nodes only (see the diagrams below)



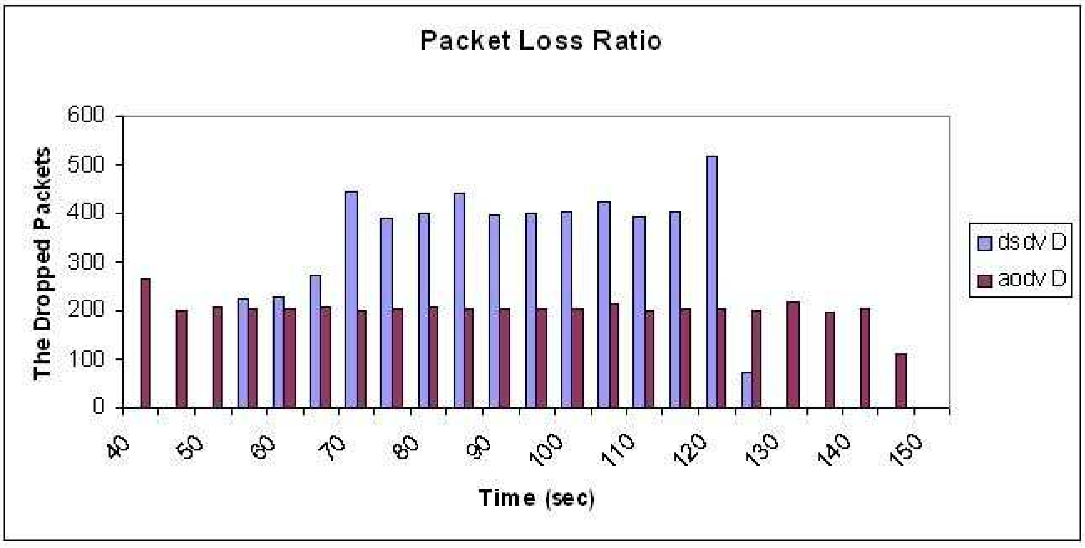
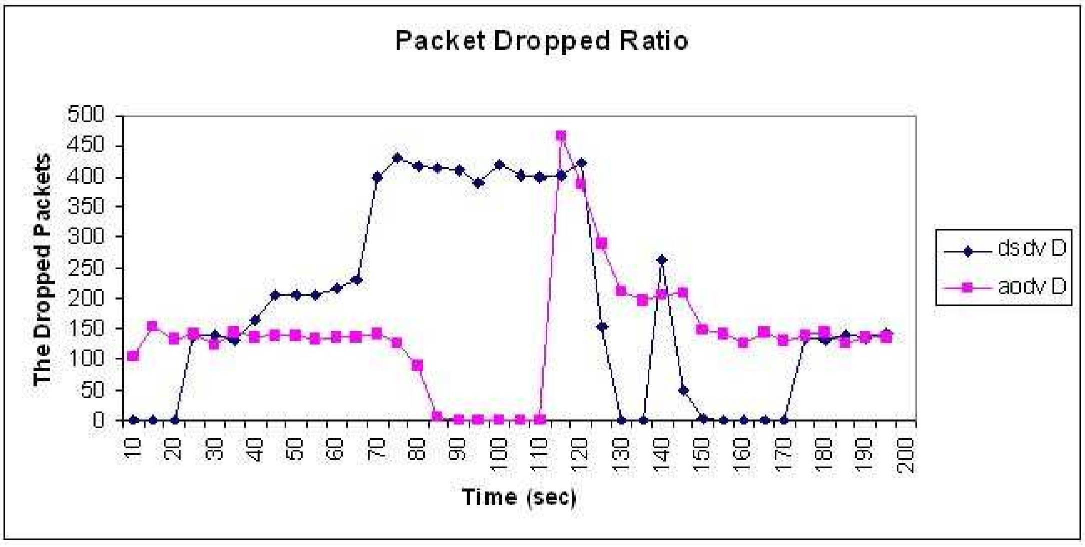


Figure 5.25 a three nodes Packet Loss Ratio

Rather than in the case of AODV routing protocol, the packets started dropping from the start (as we mentioned before that, the receiving by node 5 with AODV is higher and started before

DSDV), But in the both cases above show that, the number of packets lost was higher with DSDV routing protocol than AODV routing protocol, because the number of packets lost was higher in the case of DSDV than AODV routing protocol.

The second ad-hoc network contains 6-nodes only (see the diagrams below), and also we got the same results, the number of packets lost was higher with DSDV routing protocol than AODV routing protocol, because in the period between 85 seconds to 110 seconds contain higher level for dropping in the case of DSDV and lower case for dropping in the case of AODV.



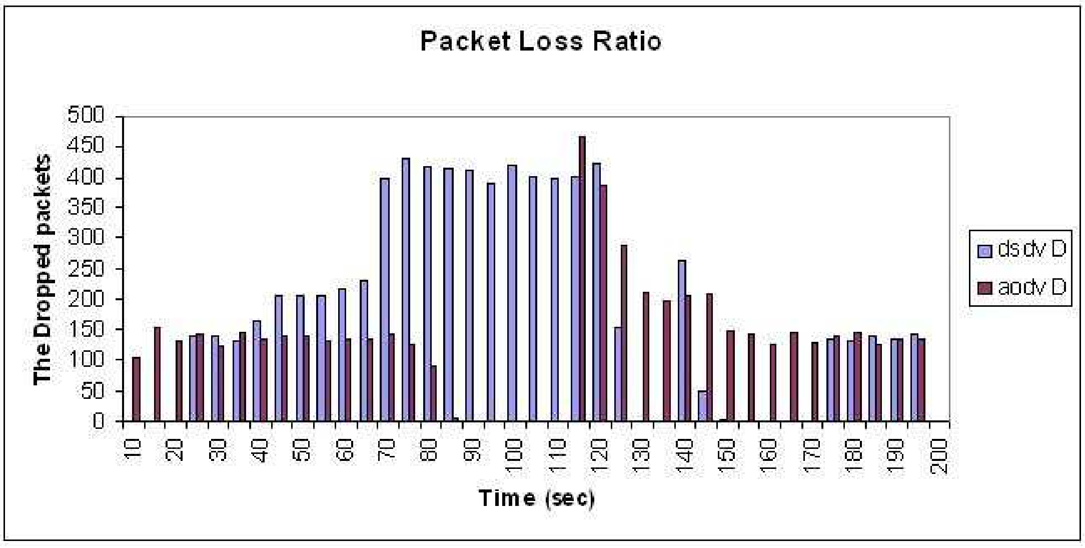


Figure 5.26 a six nodes Packet Loss Ratio

6. Summary and Conclusion

We have compared the performance of DSDV (Destination Sequenced Distance-Vector) from the Proactive family with the second type is AODV (Ad-hoc On-Demand Distance Vector) from the Reactive family. We used a detailed simulation model to demonstrate the performance characteristics of these protocols. By simulating we can argue that if delay is our main criteria than DSDV can be our best choice but if reliability and throughput are our main parameters for selection then AODV gives better results compare to others because its throughput and packet delivery ratio is best among others. While there are many other issues that need to be considered in analyzing the performance of ad-hoc networks, we believe that our work could provide intuition for future protocol selection and analysis in ad-hoc networks. While we focus only on the network throughput, reliability and the delay, it would be interesting to consider other metrics like power consumption, the number of hops to route the packet, fault tolerance, minimizing the number of control packets etc.

In the future, extensive complex simulations could be carried out to gain a more in depth performance analysis of the ad-hoc wireless networks and enhancing the performance and also for proposing new protocols and new algorithms to solve some of ad-hoc routing protocol problems.

7. References

[1] S. Murthy and J. J. Garcia-Luna-Aceves, “An Efficient Routing Protocol for Wireless Networks" ACM Mobile Networks and APP.J, Special Issue on Routing in Mobile Communication Networks, 1996.

[2] C. E. Perkins, E. M. Belding-Royer, and S. Das, “Ad hoc on-demand distance vector (AODV) routing,” RFC 3561, July 2003, Category: Experimental, work in progress.

[3] C.E. Perkins and P. Bhagwat, “Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers,” Comp. Commun. Rev., Oct. 1994.

[4] V. D. Park and M. S. Carson, “A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks,” Proc. INFOCOM ’97, Apr. 1997.

[5] C. Siva Ram Murthy and B. S. Manoj, “Ad Hoc Wireless Networks, Architectures and Protocols,” first Indian reprint 2005.

[6] L. R. Ford Jr. and D. R. Fulkerson, Flows in Networks, Princeton Univ. Press, 1962

[7] Mehran Abolhasan, Tadeusz Wysocki, Eryk Dutkiewicz "A review of Routing Protocols for Mobile Ad Hoc Networks".

[8] Imrich Chlamtac, Marco Conti, and Jennifer J. N. Liu "Mobile Ad Hoc Networking: Imperatives and challenges".

[9] Y.-H. Wang, C.M. Chung, and C.-C. Chuang "Ad Hoc Routing Protocol Setup with On-Demand Backup Node".

[10] D. D. Perkins, H. D. Hughes, and C. B. Owen "Factors Affecting the Performance of Ad Hoc Networks".