**Performance analysis and simulation of various**

**AD-HOC routing protocol**

**(DSR, DSDV, AODV)**

# J COMPONENT PROJECT

Submitted By

**SHERIL S. PHILIP-15BCB0120**

**SAMMYAK PRASHANT ROKADE-15BCB0107**

For the course

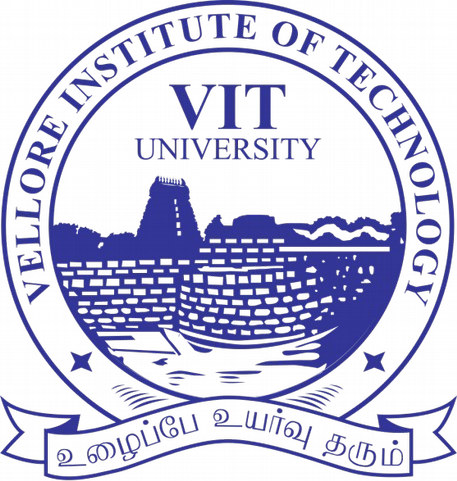
**CSE1004- Networks and communication**

UNDER THE GUIDANCE OF

**Prof. Jaisankar N**

ASSOCIATE PROFESSOR

VIT UNIVERSITY, VELLORE



# ABSTRACT

It is commonly known to the wireless research community that use of efficient routing algorithms in ad hoc networks offers a number of benefits. Some of them are: larger throughput, lower average end-to-end delay, decrement in the number of lost data packets and generally an improved network performance. This paper presents a comparative study of the Ad-hoc routing protocols. In this reprot, a comparison has been made in between different- different Ad- hoc Routing protocols (Proactive and Reactive), which are: AODV (Ad hoc on- demand Distance Vector routing), DSR (Dynamic Source Routing) and DSDV (Destination Sequence Distance Vector). The comparison have been made by using Network Simulator 2.

Keywords: MANET, AODV, DSR, DSDV, NS 2.

# 1. INTRODUCTION

An ad-hoc network is a collection of wireless mobile hosts forming a temporary network without the aid of any centralized administration [1]. Mobile Ad-hoc networks are self-configuring multihop wireless networks[3]. Wireless ad-hoc networks have gained a lot of importance in wireless communications. Because, Wireless communication is established by nodes which will act as routers. Routing in these networks is highly complex due to moving nodes and hence many protocols have been developed [2].

Our contribution is to compare the performance of most popular routing protocols in ad hoc network and the goal is to test the efficiency of the above routing protocols in scenarios.

1. **DEFINITION AND EXPLANATION**
   1. *DESTINATION SEQUENCED DISTANCE VECTOR (DSDV)*

DSDV was one of the first proactive routing protocols available for Ad Hoc networks. It was developed by C. Perkins in 1994, 5 years before the informational RFC of the MANET group. It has not been standardised by any regulation authorities but is still a reference.

* + 1. *ALGORITHM*

DSDV is based on the Bellman-Ford algorithm. First designed for graph search applications, this algorithm is also used for routing since it is the one used by RIP. With DSDV, each routing table will contain all available destinations, with the associated next hop, the associated metric (numbers of hops), and a sequence number originated by the destination node.

Tables are updated in the topology per exchange between nodes. Each node will broadcast to its neighbours entries in its 3table. This exchange of entries can be made by dumping the whole routing table, or by performing an incremental update, that means exchanging just recently updated routes. Nodes who receive this data can then update their tables if they received a better route, or a new one. Updates are performed on a regular basis, and are instantly scheduled if a new event is detected in the topology. If there are frequent changes in topology, full table exchange will be preferred whereas in a stable topology, incremental updates will cause less traffic. The route selection is performed on the metric and sequence number criteria. The sequence number is a time indication sent by the destination node. It allows the table update process, as if two identical routes are known, the one with the best sequence number is kept and used, while the other is destroyed (considered as a stale entry).

* + 1. *ILLUSTRATION*

Let us consider the two following topologies. We suppose at this time the network is stable, each node has a correct routing table of all destinations.

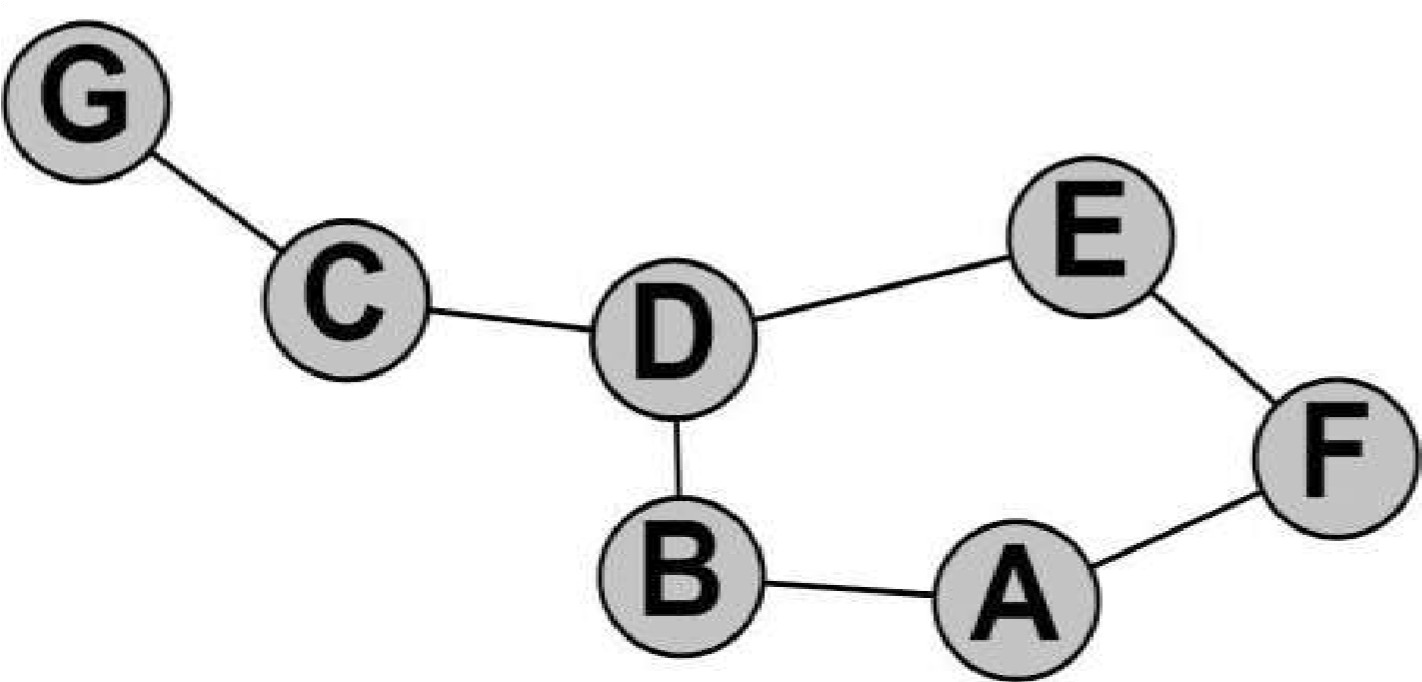


fig 2.1.1

Then, we suppose G is moving, and at t+1.

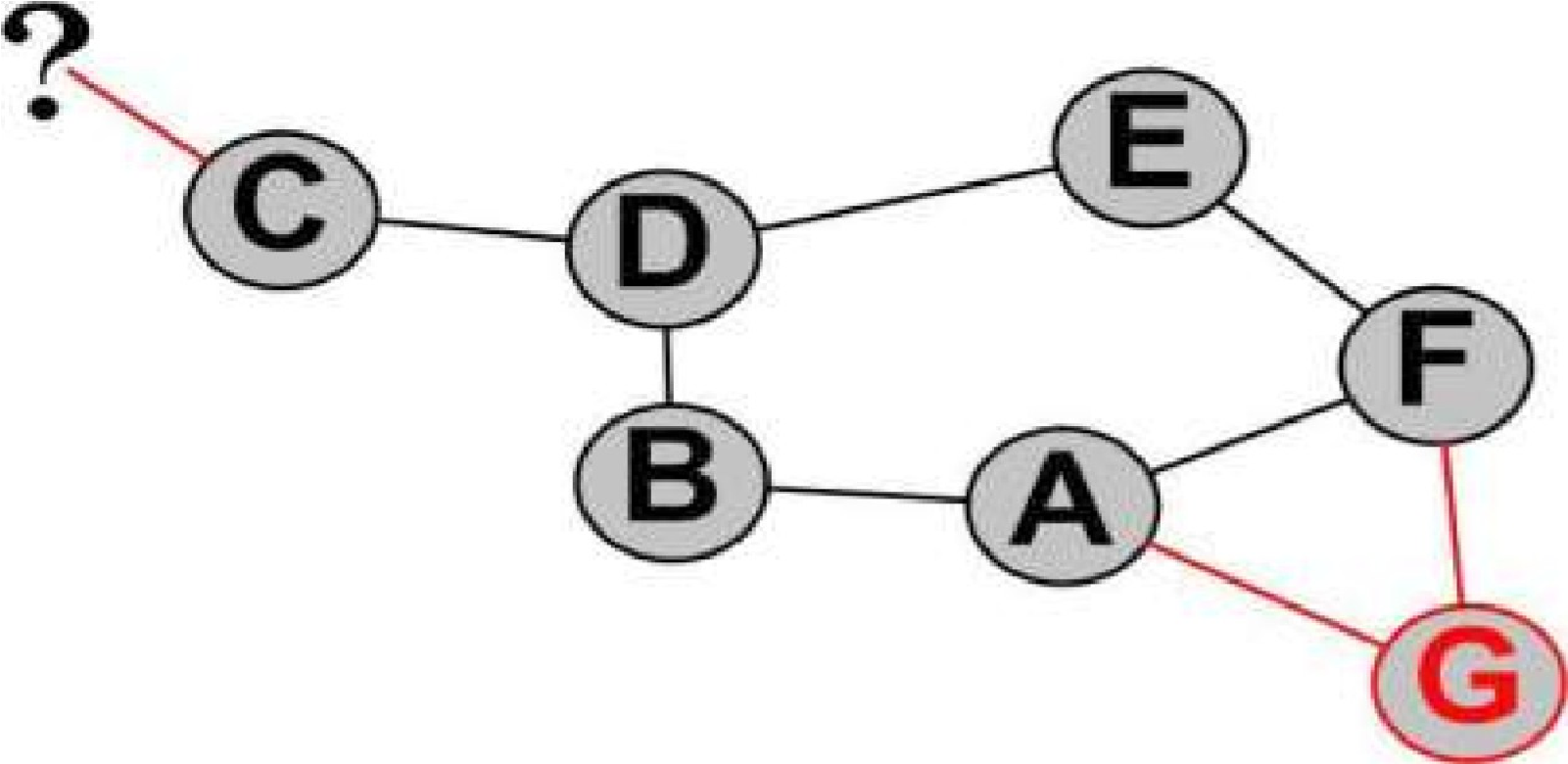


Fig 2.1

* 1. *AD HOC ON-DEMAND DISTANCE VECTOR (AODV)*

AODV was proposed to standardisation by the RFC 3561 in July 2003. It was designed by the same people who designed DSDV. AODV is a distance vector routing protocol, which means routing decisions will be taken depending on the number of hops to destination. A particularity of this network is to support both multicast and unicast routing.

* + 1. *ALGORITHM*

The AODV algorithm is inspired from the Bellman-Ford algorithm like DSDV. The principal change is to be On Demand. The node will be silent while it does not have data to send. Then, if the upper layer is requesting a route for a packet, a ―ROUTE REQUEST packet will be sent to‖ the direct neighbourhood. If a neighbour has a route corresponding to the request, a packet

―ROUTE REPLY will be returned. This packet is like a ―use me answer. Otherwise, each‖ ‖ neighbour will forward the ―ROUTE REQUEST to their own neighbourhood, except for the‖ originator and increment the hop value in the packet data. They also use this packet for building a reverse route entry (to the originator). This process occurs until a route has been found. Another part of this algorithm is the route maintenance. While a neighbour is no longer available, if it was a hop for a route, this route is not valid anymore. AODV uses ―HELLO packets on a regular basis‖ to check if they are active neighbours. Active neighbours are the ones used during a previous route discovery process. If there is no response to the ―HELLO packet sent to a node, then, the‖ originator deletes all associated routes in its routing table. ―HELLO packets are similar to ping‖ requests. While transmitting, if a link is broken (a station did not receive acknowledgment from the layer 2), a ―ROUTE ERROR packet is unicast to all previous forwarders and to the sender of the‖ packet.

* 1. ILLUSTRATION

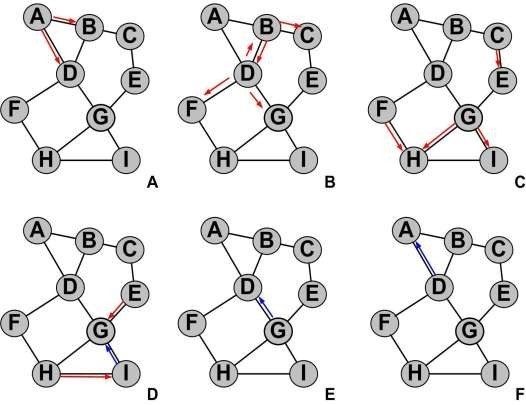


Fig 2.2

In the example illustrated by figure 2.2, A needs to send a packet to I. A ―ROUTE REQUEST‖ packet will be generated and sent to B and D (a). B and D add A in their routing table, as a reverse route, and forward the ―ROUTE REQUEST packet to their neighbours (b). B and D ignored the‖ packet they exchanged each others (as duplicates). The forwarding process continues while no route is known (c). Once I receives the ―ROUTE REQUEST from G (d), it generates the‖ ―ROUTE REPLY packet and sends it to the node it received from. Duplicate packets continue to‖ be ignored while the ―ROUTE REPLY packet goes on the shortest way to A, using previously‖ established reverse routes (e and f).

The reverse routes created by the other nodes that have not been used for the ―ROUTE REPLY‖ are deleted after a delay. G and D will add the route to I once they receive the ―ROUTE REPLY‖ packet.

2.3 *DYNAMIC SOURCE ROUTING (DSR)*

As a reactive protocol, DSR has some similitude with AODV. Thus, the difference with AODV is that DSR focuses on the source routing rather than on exchanging tables.

* + 1. *00*

DSR uses explicit source routing, which means that each time a data packet is sent, it contains the list of nodes it will use to be forwarded. In other terms, a sent packet contains the route it will use. This mechanism allows nodes on the route to cache new routes, and also, allows the originator to Sspecify the route it wants, depending on criteria such as load balancing, QoS… This mechanism also avoids routing loops.

If a node has to send a packet to another one, and it has no route for that, it initiates a route discovery process. This process is very similar to the AODV protocol as a route request is broadcast to the initiator neighbourhood until a route is found. Thus, the difference is that every node used for broadcasting this route request packet deduces the route to the originator, and keeps it in cache. Also, there can be many route replies for a single request.

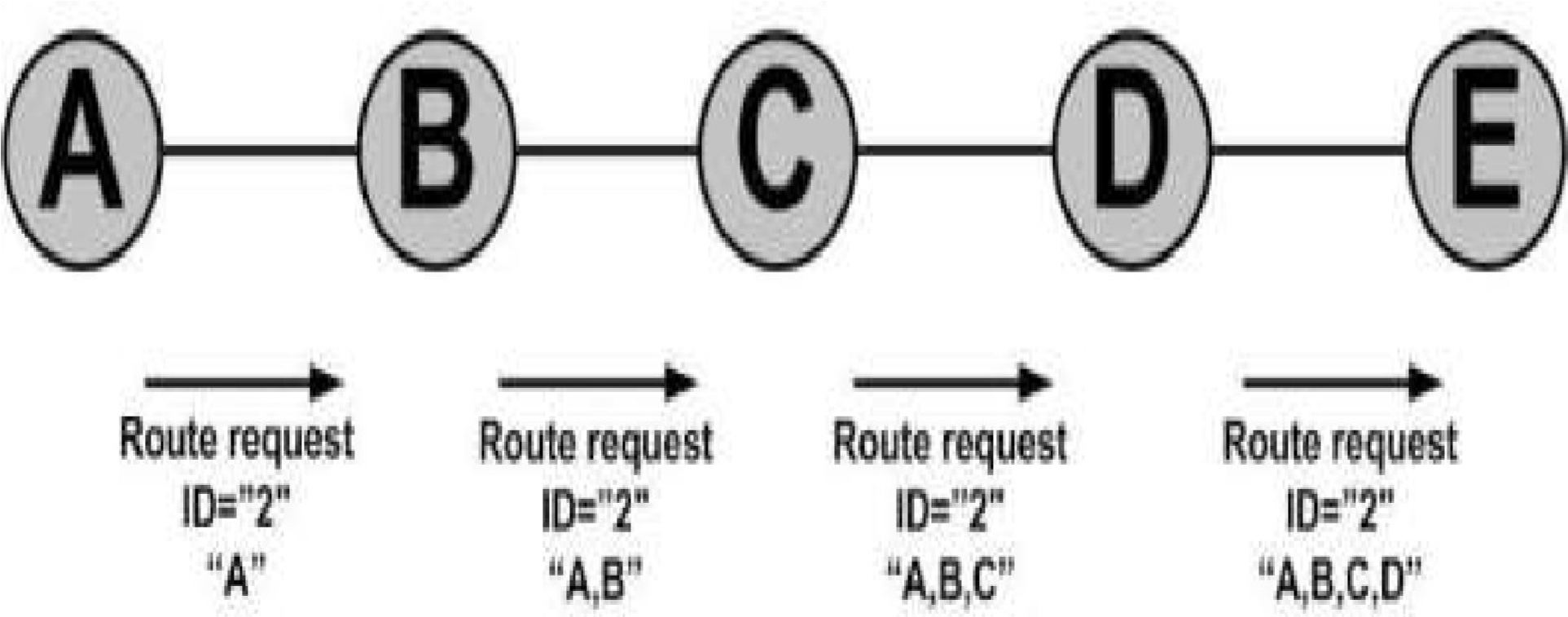


fig 2.3

In figure 2.3, A wants a route to E. It broadcasts a route request to its neighbours with an arbitrary chosen ID. Neighbours forward this broadcast, and at each node, the reverse route entry is added into the route request packet. When E receives this route request, it can sent a route reply to A using the reverse route included in the packet. The route reply packet contains the request ID and the reverse route. Another difference with AODV is in the route maintenance process. DSR does not use broadcasts such as AODV s ―HELLO packets. Instead, it uses layer two built-in‟ ‖ acknowledgments.



Fig 2.4

In Figure 2.4, A is responsible for the flow between A and B, B is responsible for the flow between B and C, and so on. If A is sending data to E, with a previously cached route, and C didn t receive‟ any acknowledgment from D, then, C deduces the link is broken and sends a

―ROUTE ERROR packet to A and any other nodes who had previously used this link.‖ Concerned nodes will then remove this route from their table, and use another one if they had other answers from their previous queries. Otherwise, the route discovery process is used in order to find another path to E.

## 3. CODE

3.1 Main NS2 code (tcl file)

set a 1 while {$a == 1 } {

puts "Enter the Routing Agents in mobile networking"

puts "1. AODV" puts "2. DSDV"

puts "3. DSR"

set top [gets stdin] if {$top == 1} {

set opt(chan) Channel/WirelessChannel ;# channel type set opt(prop) Propagation/TwoRayGround ;# radio-propagation model set opt(netif) Phy/WirelessPhy ;# network interface type set opt(mac) Mac/802\_11 ;# MAC type set opt(ifq) Queue/DropTail/PriQueue ;# interface queue type set opt(ll) LL ;# link layer type set opt(ant) Antenna/OmniAntenna ;# antenna model set opt(ifqlen) 50 ;# max packet in ifq set opt(nn) 22 ;# number of mobilenodes

set opt(rp) AODV ;# routing protocol

set opt(x) 1800 ;# X dimension of topography set opt(y) 840 ;# Y dimension of topography

### Setting The Simulator Objects

set ns\_ [new Simulator] #create the nam and trace file: set tracefd [open aodv.tr w] $ns\_ trace-all $tracefd

set namtrace [open aodv.nam w]

$ns\_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

set topo [new Topography] $topo load\_flatgrid $opt(x) $opt(y) create-god $opt(nn)

set chan\_1\_ [new $opt(chan)]

#### Setting The Distance Variables

#strength of transmitted signal vs distance # For model 'TwoRayGround' set dist(5m) 7.69113e-06 set dist(9m) 2.37381e-06 set dist(10m) 1.92278e-06 set dist(11m) 1.58908e-06 set dist(12m) 1.33527e-06 set dist(13m) 1.13774e-06 set dist(14m) 9.81011e-07 set dist(15m) 8.54570e-07 set dist(16m) 7.51087e-07 set dist(20m) 4.80696e-07 set dist(25m) 3.07645e-07 set dist(30m) 2.13643e-07 set dist(35m) 1.56962e-07 set dist(40m) 1.56962e-10 set dist(45m) 1.56962e-11 set dist(50m) 1.20174e-13

Phy/WirelessPhy set CSThresh\_ $dist(50m)

Phy/WirelessPhy set RXThresh\_ $dist(50m)

#RX-THRESH-reciever sensitivity transmission range

# CSTHRESH-carrier sense transmission range

# Defining Node Configuration

$ns\_ node-config -adhocRouting $opt(rp) \

-llType $opt(ll) \

-macType $opt(mac) \

-ifqType $opt(ifq) \

-ifqLen $opt(ifqlen) \

-antType $opt(ant) \

-propType $opt(prop) \

-phyType $opt(netif) \

-topoInstance $topo \

-agentTrace ON \

-routerTrace ON \

-macTrace ON \

-movementTrace ON \

-energyModel "EnergyModel" \

-idlePower 1.0 \

-rxPower 1.0 \

-txPower 1.0 \

-sleepPower 0.001 \

-transitionPower 0.2 \

-transitionTime 0.005 \

-initialEnergy 1000 \

-channel $chan\_1\_

### Creating The WIRELESS NODES

set Server1 [$ns\_ node] set Server2 [$ns\_ node] set n2 [$ns\_ node] set n3 [$ns\_ node] set n4 [$ns\_ node] set n5 [$ns\_ node] set n6 [$ns\_ node] set n7 [$ns\_ node] set n8 [$ns\_ node] set n9 [$ns\_ node] set n10 [$ns\_ node] set n11 [$ns\_ node] set n12 [$ns\_ node] set n13 [$ns\_ node] set n14 [$ns\_ node] set n15 [$ns\_ node] set n16 [$ns\_ node] set n17 [$ns\_ node] set n18 [$ns\_ node] set n19 [$ns\_ node] set n20 [$ns\_ node] set n21 [$ns\_ node] set n22 [$ns\_ node]

set opt(seed) 0.1 set a [ns-random $opt(seed)]

set i 0 while {$i < 5} { incr i }

# opt(seed)=seeding/generating random number in same way as we do in back-off algorithm at mac layer.

### Setting The Initial Positions of Nodes

$Server1 set X\_ 513.0

$Server1 set Y\_ 517.0

$Server1 set Z\_ 0.0

$Server2 set X\_ 1445.0

$Server2 set Y\_ 474.0

$Server2 set Z\_ 0.0

$n2 set X\_ 36.0

$n2 set Y\_ 529.0 $n2 set Z\_ 0.0

$n3 set X\_ 143.0

$n3 set Y\_ 666.0 $n3 set Z\_ 0.0

$n4 set X\_ 201.0

$n4 set Y\_ 552.0

$n4 set Z\_ 0.0

$n5 set X\_ 147.0

$n5 set Y\_ 403.0

$n5 set Z\_ 0.0

$n6 set X\_ 230.0

$n6 set Y\_ 291.0

$n6 set Z\_ 0.0

$n7 set X\_ 295.0 $n7 set Y\_ 419.0 $n7 set Z\_ 0.0

$n8 set X\_ 363.0 $n8 set Y\_ 335.0 $n8 set Z\_ 0.0

$n9 set X\_ 334.0 $n9 set Y\_ 647.0 $n9 set Z\_ 0.0

$n10 set X\_ 304.0 $n10 set Y\_ 777.0

$n10 set Z\_ 0.0

$n11 set X\_ 412.0 $n11 set Y\_ 194.0

$n11 set Z\_ 0.0

$n12 set X\_ 519.0 $n12 set Y\_ 361.0 $n12 set Z\_ 0.0

$n13 set X\_ 569.0 $n13 set Y\_ 167.0 $n13 set Z\_ 0.0

$n14 set X\_ 349.0 $n14 set Y\_ 546.0 $n14 set Z\_ 0.0

$n15 set X\_ 466.0 $n15 set Y\_ 668.0 $n15 set Z\_ 0.0

$n16 set X\_ 489.0 $n16 set Y\_ 794.0 $n16 set Z\_ 0.0

$n17 set X\_ 606.0

$n17 set Y\_ 711.0 $n17 set Z\_ 0.0

$n18 set X\_ 630.0 $n18 set Y\_ 626.0 $n18 set Z\_ 0.0

$n19 set X\_ 666.0 $n19 set Y\_ 347.0 $n19 set Z\_ 0.0

$n20 set X\_ 641.0 $n20 set Y\_ 152.0 $n20 set Z\_ 0.0

$n21 set X\_ 882.0 $n21 set Y\_ 264.0

$n21 set Z\_ 0.0

$n22 set X\_ 761.0 $n22 set Y\_ 441.0

$n22 set Z\_ 0.0

## Giving Mobility to Nodes

$ns\_ at 0.75 "$n2 setdest 379.0 349.0 20.0"

$ns\_ at 0.75 "$n3 setdest 556.0 302.0 20.0" $ns\_ at 0.20 "$n4 setdest 309.0 211.0 20.0" $ns\_ at 1.25 "$n5 setdest 179.0 333.0 20.0"

$ns\_ at 0.75 "$n6 setdest 139.0 63.0 20.0"

$ns\_ at 0.75 "$n7 setdest 320.0 27.0 20.0"

$ns\_ at 1.50 "$n8 setdest 505.0 124.0 20.0"

$ns\_ at 1.25 "$n9 setdest 274.0 487.0 20.0"

$ns\_ at 1.25 "$n10 setdest 494.0 475.0 20.0"

$ns\_ at 1.25 "$n11 setdest 899.0 757.0 25.0" $ns\_ at 0.50 "$n12 setdest 598.0 728.0 25.0"

$ns\_ at 0.25 "$n13 setdest 551.0 624.0 25.0"

$ns\_ at 1.25 "$n14 setdest 397.0 647.0 25.0"

$ns\_ at 1.25 "$n15 setdest 748.0 688.0 25.0"

$ns\_ at 1.25 "$n16 setdest 842.0 623.0 25.0"

$ns\_ at 1.25 "$n17 setdest 678.0 548.0 25.0"

$ns\_ at 0.75 "$n18 setdest 741.0 809.0 20.0"

$ns\_ at 0.75 "$n19 setdest 437.0 699.0 20.0"

$ns\_ at 0.20 "$n20 setdest 159.0 722.0 50.0"

$ns\_ at 0.25 "$n21 setdest 900.0 250.0 50.0"

$ns\_ at 0.20 "$n22 setdest 839.0 144.0 50.0"

## Setting The Node Size

$ns\_ initial\_node\_pos $Server1 125

$ns\_ initial\_node\_pos $Server2 125

$ns\_ initial\_node\_pos $n2 70

$ns\_ initial\_node\_pos $n3 70

$ns\_ initial\_node\_pos $n4 40

$ns\_ initial\_node\_pos $n5 70

$ns\_ initial\_node\_pos $n6 70

$ns\_ initial\_node\_pos $n7 70

$ns\_ initial\_node\_pos $n8 70

$ns\_ initial\_node\_pos $n9 70

$ns\_ initial\_node\_pos $n10 70 $ns\_ initial\_node\_pos $n11 70

$ns\_ initial\_node\_pos $n12 70

$ns\_ initial\_node\_pos $n13 70

$ns\_ initial\_node\_pos $n14 70

$ns\_ initial\_node\_pos $n15 70

$ns\_ initial\_node\_pos $n16 70

$ns\_ initial\_node\_pos $n17 70

$ns\_ initial\_node\_pos $n18 70

$ns\_ initial\_node\_pos $n19 70

$ns\_ initial\_node\_pos $n20 70

$ns\_ initial\_node\_pos $n21 70

$ns\_ initial\_node\_pos $n22 70

#### Setting The Labels For Nodes

$ns\_ at 0.0 "$Server1 label Server1"

$ns\_ at 0.0 "$Server2 label Server2"

$ns\_ at 0.0 "$n2 label node2"

$ns\_ at 0.0 "$n3 label node3"

$ns\_ at 0.0 "$n4 label node4"

$ns\_ at 0.0 "$n5 label node5"

$ns\_ at 0.0 "$n6 label node6"

$ns\_ at 0.0 "$n7 label node7"

$ns\_ at 0.0 "$n8 label node8"

$ns\_ at 0.0 "$n9 label node9"

$ns\_ at 0.0 "$n10 label node10"

$ns\_ at 0.0 "$n11 label node11"

$ns\_ at 0.0 "$n12 label node12"

$ns\_ at 0.0 "$n13 label node13"

$ns\_ at 0.0 "$n14 label node14"

$ns\_ at 0.0 "$n15 label node15"

$ns\_ at 0.0 "$n16 label node16"

$ns\_ at 0.0 "$n17 label node17"

$ns\_ at 0.0 "$n18 label node18"

$ns\_ at 0.0 "$n19 label node19"

$ns\_ at 0.0 "$n20 label node20"

$ns\_ at 0.0 "$n20 label node21"

$ns\_ at 0.0 "$n22 label node22"

$n2 color green

$ns\_ at 0.0 "$n2 color green"

$n3 color green

$ns\_ at 0.0 "$n3 color green"

$n4 color green

$ns\_ at 0.0 "$n4 color green"

$n5 color green

$ns\_ at 0.0 "$n5 color green"

$n6 color green

$ns\_ at 0.0 "$n6 color green"

$n7 color green

$ns\_ at 0.0 "$n7 color green"

$n8 color green

$ns\_ at 0.0 "$n8 color green"

$n9 color yellow

$ns\_ at 0.0 "$n9 color yellow"

$n10 color yellow

$ns\_ at 0.0 "$n10 color yellow"

$n11 color yellow

$ns\_ at 0.0 "$n11 color yellow"

$n12 color pink

$ns\_ at 0.0 "$n12 color pink"

$n13 color pink

$ns\_ at 0.0 "$n13 color pink"

$n14 color pink

$ns\_ at 0.0 "$n14 color pink"

$n15 color pink

$ns\_ at 0.0 "$n15 color pink"

$n16 color pink

$ns\_ at 0.0 "$n16 color pink"

$n17 color orange

$ns\_ at 0.0 "$n17 color orange"

$n18 color orange

$ns\_ at 0.0 "$n18 color orange" $n19 color orange

$ns\_ at 0.0 "$n19 color orange"

$n20 color orange

$ns\_ at 0.0 "$n20 color orange"

$n21 color orange

$ns\_ at 0.0 "$n21 color orange"

$n22 color orange

$ns\_ at 0.0 "$n22 color orange"

$Server1 color maroon

$ns\_ at 0.0 "$Server1 color maroon"

$Server2 color maroon

$ns\_ at 0.0 "$Server2 color maroon"

## SETTING ANIMATION RATE

$ns\_ at 0.0 "$ns\_ set-animation-rate 12.5ms"

# COLORING THE NODES

$n9 color blue

$ns\_ at 4.71 "$n9 color blue"

$n5 color blue

$ns\_ at 7.0 "$n5 color blue"

$n2 color blue

$ns\_ at 7.29 "$n2 color blue"

$n16 color blue

$ns\_ at 7.59 "$n16 color blue"

$n9 color maroon

$ns\_ at 7.44 "$n9 color maroon"

$ns\_ at 7.43 "$n9 label TTLover"

$ns\_ at 7.55 "$n9 label \"\""

$n12 color blue

$ns\_ at 7.85 "$n12 color blue"

#### Establishing Communication

set udp0 [$ns\_ create-connection UDP $Server1 LossMonitor $n18 0]

#Loss Monitor is attached with receiver node.Loss Monitor objects trace #out the lost packets, and received packets details.Packet Loss is #measured by accessing the loss monitor object.

$udp0 set fid\_ 1

#id field is used for specifying stream color for the NAM display set cbr0 [$udp0 attach-app Traffic/CBR]

$cbr0 set packetSize\_ 1000

$cbr0 set interopt\_ .07

$ns\_ at 0.0 "$cbr0 start"

$ns\_ at 4.2 "$cbr0 stop"

set udp1 [$ns\_ create-connection UDP $Server1 LossMonitor $n22 0]

$udp1 set fid\_ 1

set cbr1 [$udp1 attach-app Traffic/CBR]

$cbr1 set packetSize\_ 1000

$cbr1 set interopt\_ .07

$ns\_ at 0.1 "$cbr1 start"

$ns\_ at 7.1 "$cbr1 stop"

set udp2 [$ns\_ create-connection UDP $n21 LossMonitor $n20 0]

$udp2 set fid\_ 1

set cbr2 [$udp2 attach-app Traffic/CBR]

$cbr2 set packetSize\_ 1000

$cbr2 set interopt\_ .07

$ns\_ at 0.4 "$cbr2 start"

$ns\_ at 6.1 "$cbr2 stop"

set udp3 [$ns\_ create-connection UDP $Server1 LossMonitor $n15 0]

$udp3 set fid\_ 1

set cbr3 [$udp3 attach-app Traffic/CBR]

$cbr3 set packetSize\_ 1000

$cbr3 set interopt\_ 5

$ns\_ at 4.0 "$cbr3 start"

$ns\_ at 4.1 "$cbr3 stop"

set udp4 [$ns\_ create-connection UDP $Server1 LossMonitor $n14 0]

$udp4 set fid\_ 1

set cbr4 [$udp4 attach-app Traffic/CBR]

$cbr4 set packetSize\_ 1000

$cbr4 set interopt\_ 5

$ns\_ at 4.0 "$cbr4 start"

$ns\_ at 4.1 "$cbr4 stop"

set udp5 [$ns\_ create-connection UDP $n15 LossMonitor $n16 0]

$udp5 set fid\_ 1

set cbr5 [$udp5 attach-app Traffic/CBR]

$cbr5 set packetSize\_ 1000

$cbr5 set interopt\_ 5

$ns\_ at 4.0 "$cbr5 start"

$ns\_ at 4.1 "$cbr5 stop"

set udp6 [$ns\_ create-connection UDP $n15 LossMonitor $n17 0]

$udp6 set fid\_ 1

set cbr6 [$udp6 attach-app Traffic/CBR]

$cbr6 set packetSize\_ 1000

$cbr6 set interopt\_ 5

$ns\_ at 4.0 "$cbr6 start"

$ns\_ at 4.1 "$cbr6 stop"

set udp7 [$ns\_ create-connection UDP $n14 LossMonitor $n4 0]

$udp7 set fid\_ 1

set cbr7 [$udp7 attach-app Traffic/CBR]

$cbr7 set packetSize\_ 1000

$cbr7 set interopt\_ 5

$ns\_ at 4.0 "$cbr7 start"

$ns\_ at 4.1 "$cbr7 stop"

set udp8 [$ns\_ create-connection UDP $n14 LossMonitor $n9 0]

$udp8 set fid\_ 1

set cbr8 [$udp8 attach-app Traffic/CBR]

$cbr8 set packetSize\_ 1000

$cbr8 set interopt\_ 5

$ns\_ at 4.0 "$cbr8 start"

$ns\_ at 4.1 "$cbr8 stop"

set udp9 [$ns\_ create-connection UDP $n4 LossMonitor $n3 0]

$udp9 set fid\_ 1

set cbr9 [$udp9 attach-app Traffic/CBR]

$cbr9 set packetSize\_ 1000

$cbr9 set interopt\_ 5

$ns\_ at 4.0 "$cbr9 start"

$ns\_ at 4.1 "$cbr9 stop"

set udp10 [$ns\_ create-connection UDP $n4 LossMonitor $n2 0]

$udp10 set fid\_ 1

set cbr10 [$udp10 attach-app Traffic/CBR]

$cbr10 set packetSize\_ 1000

$cbr10 set interopt\_ 5

$ns\_ at 4.0 "$cbr10 start"

$ns\_ at 4.1 "$cbr10 stop"

set udp11 [$ns\_ create-connection UDP $n9 LossMonitor $n16 0]

$udp11 set fid\_ 1

set cbr11 [$udp11 attach-app Traffic/CBR]

$cbr11 set packetSize\_ 1000

$cbr11 set interopt\_ 5

$ns\_ at 4.0 "$cbr11 start"

$ns\_ at 4.1 "$cbr11 stop"

set udp12 [$ns\_ create-connection UDP $n9 LossMonitor $n10 0]

$udp12 set fid\_ 1

set cbr12 [$udp12 attach-app Traffic/CBR]

$cbr12 set packetSize\_ 1000

$cbr12 set interopt\_ 5

$ns\_ at 4.0 "$cbr12 start"

$ns\_ at 4.1 "$cbr12 stop"

#ANNOTATIONS DETAILS

$ns\_ at 0.0 "$ns\_ trace-annotate \"MOBILE NODE MOVEMENTS\""

$ns\_ at 4.1 "$ns\_ trace-annotate \"NODE27 CACHE THE DATA FRO SERVER\""

#$ns\_ at 4.59 "$ns\_ trace-annotate \"PACKET LOSS AT NODE27\""

$ns\_ at 4.71 "$ns\_ trace-annotate \"NODE10 CACHE THE DATA\""

### PROCEDURE TO STOP

proc stop {} {

global ns\_ tracefd global ns\_ namtrace $ns\_ flush-trace close $tracefd close $namtrace

exec nam aodv.nam & exit 0

}

puts "Starting Simulation........"

$ns\_ at 25.0 "stop"

$ns\_ run

} elseif {$top == 2} { # 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) 7 ;# number of mobilenodes

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) 100 ;# time of simulation end

set ns [new Simulator]

set tracefd [open dsdv.tr w] set namtrace [open dsdv.nam w]

$ns use-newtrace

$ns trace-all $tracefd

$ns namtrace-all-wireless $namtrace $val(x) $val(y)

Antenna/OmniAntenna set X\_ 0

Antenna/OmniAntenna set Y\_ 0

Antenna/OmniAntenna set Z\_ 1.5

Antenna/OmniAntenna set Gt\_ 1.0

Antenna/OmniAntenna set Gr\_ 1.0

# 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 OFF \

-movementTrace ON \

-energyModel "EnergyModel" \

-rxPower 1.0 \

-txPower 1.0 \

-initialEnergy 1000 \

-sleepPower .5 \

-transitionPower .2 \

-transitionTime .001 \

-idlePower .1

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\_ 490.0 $node\_(1) set Y\_ 285.0

$node\_(1) set Z\_ 0.0

$node\_(2) set X\_ 150.0 $node\_(2) set Y\_ 240.0

$node\_(2) set Z\_ 0.0

$node\_(3) set X\_ 250.0 $node\_(3) set Y\_ 240.0

$node\_(3) set Z\_ 0.0

$node\_(4) set X\_ 180.0

$node\_(4) set Y\_ 70.0 $node\_(4) set Z\_ 0.0

$node\_(5) set X\_ 100.0

$node\_(5) set Y\_ 70.0

$node\_(5) set Z\_ 0.0

$node\_(6) set X\_ 380.0

$node\_(6) set Y\_ 70.0

$node\_(6) set Z\_ 0.0

# Generation of movements

$ns at 10.0 "$node\_(0) setdest 250.0 250.0 10.0"

$ns at 15.0 "$node\_(1) setdest 45.0 285.0 10.0"

$ns at 29.0 "$node\_(2) setdest 480.0 300.0 10.0"

$ns at 70.0 "$node\_(3) setdest 180.0 30.0 10.0"

$ns at 80.0 "$node\_(4) setdest 80.0 30.0 10.0"

$ns at 90.0 "$node\_(5) setdest 98.0 30.0 10.0"

$ns at 80.0 "$node\_(6) setdest 50.0 30.0 10.0"

# Set a TCP connection between node\_(0) and node\_(1) set tcp [new Agent/TCP] set sink [new Agent/TCPSink] $ns attach-agent $node\_(0) $tcp

$ns attach-agent $node\_(1) $sink $ns connect $tcp $sink set ftp [new Application/FTP] $ftp attach-agent $tcp

$ns at 10.0 "$ftp start"

# Set a TCP connection between node\_(0) and node\_(2) set tcp [new Agent/TCP] set sink [new Agent/TCPSink] $ns attach-agent $node\_(0) $tcp

$ns attach-agent $node\_(2) $sink $ns connect $tcp $sink set ftp [new Application/FTP] $ftp attach-agent $tcp

$ns at 10.0 "$ftp start"

# Set a TCP connection between node\_(0) and node\_(3) set tcp [new Agent/TCP]

set sink [new Agent/TCPSink] $ns attach-agent $node\_(0) $tcp

$ns attach-agent $node\_(3) $sink $ns connect $tcp $sink set ftp [new Application/FTP] $ftp attach-agent $tcp

$ns at 10.0 "$ftp start"

# Set a TCP connection between node\_(0) and node\_(4) set tcp [new Agent/TCP] set sink [new Agent/TCPSink] $ns attach-agent $node\_(0) $tcp

$ns attach-agent $node\_(4) $sink $ns connect $tcp $sink set ftp [new Application/FTP] $ftp attach-agent $tcp

$ns at 10.0 "$ftp start"

# Set a TCP connection between node\_(0) and node\_(5) set tcp [new Agent/TCP] 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"

# Set a TCP connection between node\_(0) and node\_(6) set tcp [new Agent/TCP] set sink [new Agent/TCPSink] $ns attach-agent $node\_(0) $tcp

$ns attach-agent $node\_(6) $sink $ns connect $tcp $sink set ftp [new Application/FTP] $ftp attach-agent $tcp

$ns at 10.0 "$ftp start"

# 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

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "stop"

$ns at 150.01 "puts \"end simulation\" ; $ns halt" proc stop {} {

global ns tracefd namtrace

$ns flush-trace close $tracefd close $namtrace

#Execute nam on the trace file

exec nam dsdv.nam &

exit 0

}

$ns run

} elseif {$top == 3} {

set opt(chan) Channel/WirelessChannel ;# channel type

set opt(prop) Propagation/TwoRayGround ;# radio-propagation model

set opt(netif) Phy/WirelessPhy ;# network interface type set opt(mac) Mac/802\_11 ;# MAC type set opt(ifq) CMUPriQueue ;# interface queue type this type of queue sends packt

accord to priority(manages)

set opt(ll) LL ;# link layer type set opt(ant) Antenna/OmniAntenna ;# antenna model set opt(ifqlen) 50 ;# max packet in ifq set opt(nn) 3 ;# number of mobilenodes set opt(rp) DSR ;# routing protocol

set opt(x) 500 ;# X dimension of topography set opt(y) 400 ;# Y dimension of topography

set opt(stop) 150 ; set ns [new Simulator] #Creating trace file and nam file set tracefd [open Dsr.tr w] set namtrace [open dsr.nam w]

$ns trace-all $tracefd

$ns namtrace-all-wireless $namtrace $opt(x) $opt(y)

# set up topography object set topo [new Topography] $topo load\_flatgrid $opt(x) $opt(y) create-god $opt(nn)

# configure the nodes

$ns node-config -adhocRouting $opt(rp) \

-llType $opt(ll) \

-macType $opt(mac) \

-ifqType $opt(ifq) \

-ifqLen $opt(ifqlen) \

-antType $opt(ant) \

-propType $opt(prop) \

-phyType $opt(netif) \

-channelType $opt(chan) \

-topoInstance $topo \

-agentTrace ON \

-routerTrace ON \

-macTrace OFF \

-movementTrace ON \

-energyModel "EnergyModel" \

-idlePower 1.0 \

-rxPower 1.0 \

-txPower 1.0 \

-sleepPower 0.001 \

-transitionPower 0.2 \

-transitionTime 0.005 \

-initialEnergy 1000

for {set i 0} {$i < $opt(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\_ 490.0 $node\_(1) set Y\_ 285.0 $node\_(1) set Z\_ 0.0

$node\_(2) set X\_ 150.0 $node\_(2) set Y\_ 240.0 $node\_(2) set Z\_ 0.0

# Generation of movements

$ns at 10.0 "$node\_(0) setdest 250.0 250.0 3.0"

$ns at 15.0 "$node\_(1) setdest 45.0 285.0 5.0"

$ns at 110.0 "$node\_(0) setdest 480.0 300.0 5.0"

# 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 set ftp [new Application/FTP] $ftp attach-agent $tcp

$ns at 10.0 "$ftp start"

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

$ns at $opt(stop) "$node\_($i) reset"; }

# ending nam and the simulation

$ns at $opt(stop) "$ns nam-end-wireless $opt(stop)"

$ns at $opt(stop) "stop"

$ns at 150.01 "puts \"end simulation\" ; $ns halt" proc stop {} { global ns tracefd namtrace $ns flush-trace close $tracefd close $namtrace

exec nam dsr.nam & exit 0 }

$ns run

}

puts "want to continue (0/1)"

set a [gets stdin]

}

3.2 Analysis of trace file

3.2.1For analysis of Average throughput

BEGIN {

recvdSize = 0; startTime = 1e6; stopTime = 0;

}

{

# Trace line format: new if ($2 == "-t") { event = $1; time = $3; node\_id = $5; flow\_id = $39; pkt\_id = $41; pkt\_size = $37; flow\_t = $45; level = $19;

}

# Store start time

if (level == "AGT" && (event == "+" || event == "s") && pkt\_size >= 512) { if (time < startTime) {

startTime = time;

}

}

# Update total received packets' size and store packets arrival time if (level == "AGT" && event == "r" && pkt\_size >= 512) { if (time > stopTime) {

stopTime = time;

}

# Rip off the header hdr\_size = pkt\_size % 512; pkt\_size -= hdr\_size; # Store received packet's size

recvdSize += pkt\_size;

}

}

END { printf("AverageThroughput[kbps]=%.2f\t\tStartTime=%.2f\tStopTime=%.2f\n",(recvdSize/ (stopTimestartTime))\*(8/1000),startTime,stopTime) }

3.2.2 For analysis of jitter

BEGIN {

num\_recv=0

}

{

# Trace line format: new if ($2 == "-t") { event = $1 time = $3 node\_id = $5 flow\_id = $39 pkt\_id = $41 pkt\_size = $37 flow\_t = $45 level = $19

}

# Store packets send time if (level == "AGT" && sendTime[pkt\_id] == 0 && (event == "+" || event == "s") && pkt\_size

>= 512) { sendTime[pkt\_id] = time

}

# Store packets arrival time

if (level == "AGT" && event == "r" && pkt\_size >= 512) { recvTime[pkt\_id] = time num\_recv++

}

}

END {

# Compute average jitter jitter1 = jitter2 = tmp\_recv = 0 prev\_time = delay = prev\_delay = processed = 0

prev\_delay = -1

for (i=0; processed<num\_recv; i++) {

if(recvTime[i] != 0) { tmp\_recv++ if(prev\_time != 0) { delay = recvTime[i] - prev\_time e2eDelay = recvTime[i] - sendTime[i]

if(delay < 0) delay = 0 if(prev\_delay != -1) { jitter1 += abs(e2eDelay - prev\_e2eDelay)

jitter2 += abs(delay-prev\_delay)

}

prev\_delay = delay

prev\_e2eDelay = e2eDelay

}

prev\_time = recvTime[i]

}

processed++

}

}

END {

printf("Mean Jitter = %.2f\n",jitter1\*1000/tmp\_recv); printf("Current Jitter = %.2f\n",jitter2\*1000/tmp\_recv);

}

function abs(value) { if (value < 0) value = 0-value return value

}

3.2.3 For analysis of packet overheads and path details

#!/usr/bin/perl

use :strict;

if($#ARGV<0){

printf("Usage: <trace-file>\n");

exit 1;

}

# to open the given trace file

open(Trace, $ARGV[0]) or die "Cannot open the trace file"; my $sc = 0; # sending counter my $rc = 0; # receiving counter my $rp = 0; my $mc =0;

my %pkt\_fc = (); #packet forwarding counter while(<Trace>){ # read one line in from the file my @line = split; #split the line with delimin as space

if($line[3] eq "AGT" && $line[6] eq "cbr"){ # an application agent trace line if($line[0] eq "s"){ # a packet sent by some data source

$sc++;

$pkt\_fc{$line[5]} = 0; #define the forwarding couter for this pkt

}

if($line[0] eq "r"){ # a packet received by sink

$rc++;

$pkt\_fc{$line[5]}++; # one more hop to complement

} }

if($line[3] eq "MAC" && $line[6] eq "cbr")

{

if ($line[0] eq "s")

{ $mc++} }

if($line[3] eq "RTR"){ # a routing agent trace lineOD if($line[0] eq "f")

{

$rp++;

$pkt\_fc{$line[5]}++;

}

}

}

close(Trace); #close the file my $temp\_rc =0; my $pkt = 0; foreach $pkt ( keys %pkt\_fc ) {

$temp\_rc += $pkt\_fc{$pkt}; #the total forwarding times

}

if($rc > 0)

{

printf("sent packets= %d\n",$sc); printf("received packets= %d\n",$rc); printf("packets dropped=%d\n",$sc-$rc); printf("routing agents=%d/n",$rp); printf("routing overhead= %f\n",$rp/$sc\*100); printf("Packet delivery ratio %f\n",$rc/$sc);

printf("Average path length %f\n", $temp\_rc/$rc);

}

# 4. SIMULATION OBSERVATION

For this, we are using the *Network Simulator 2* because it is too fast specially for a large number of nodes. We used it to simulate three Routing Protocols: AODV, DSR and DSDV.

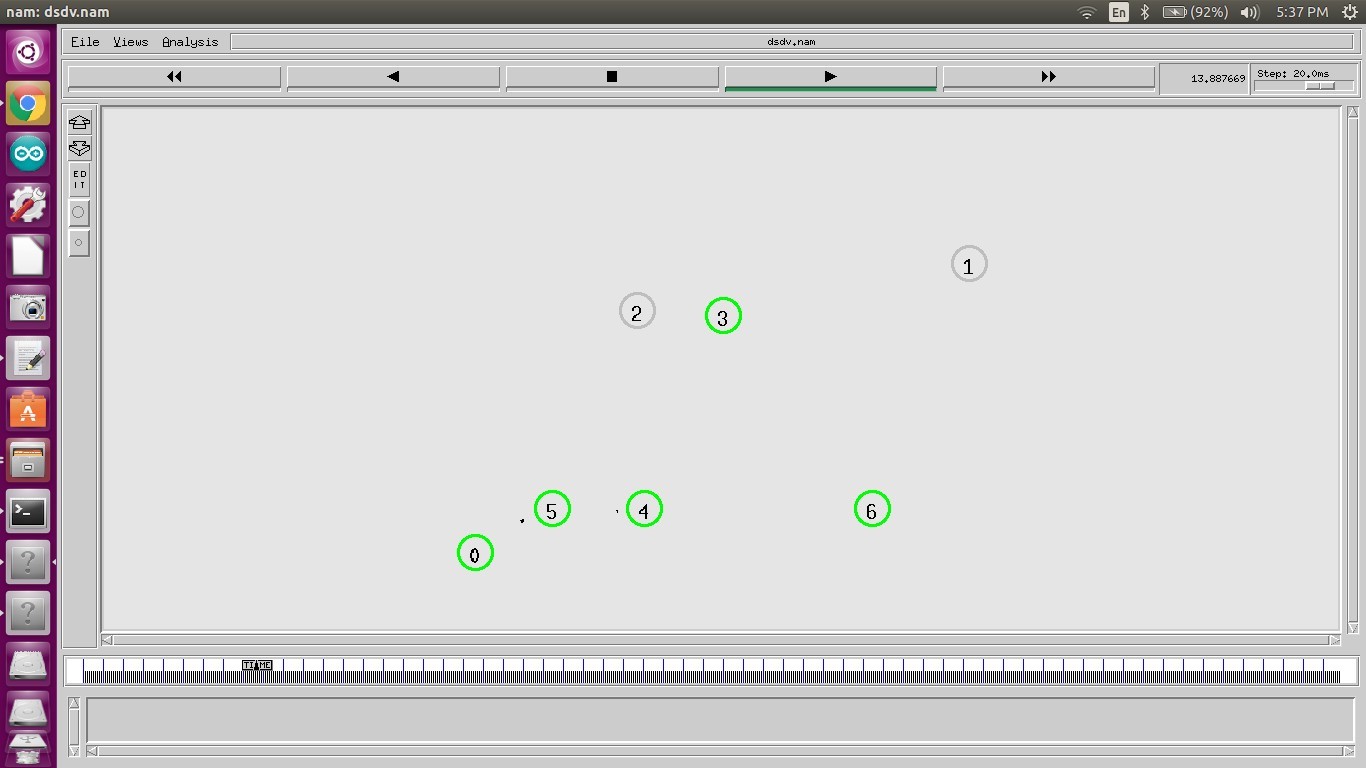


Fig 4.1 (DSDV)

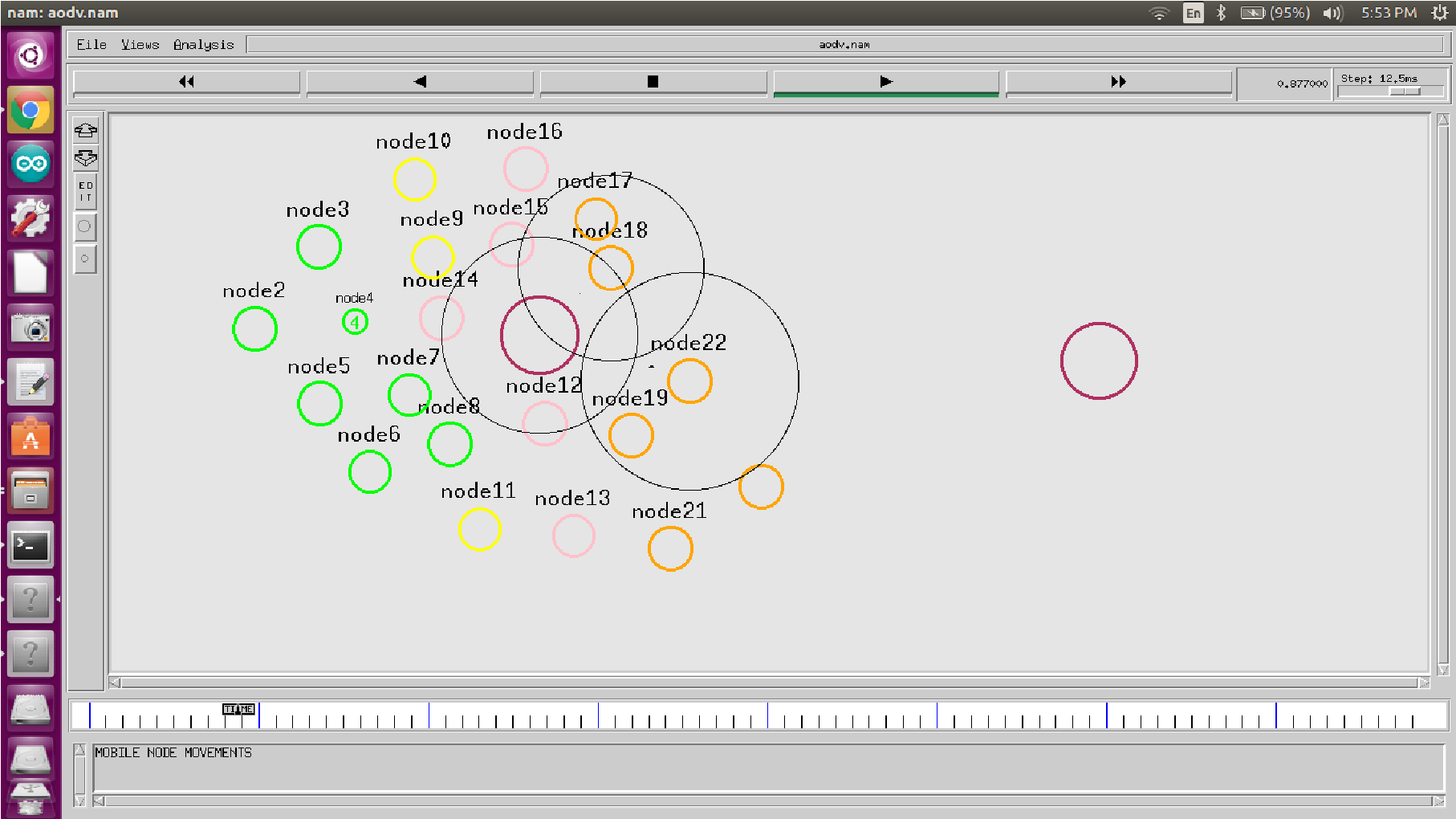


Fig 4.2 (AODV)

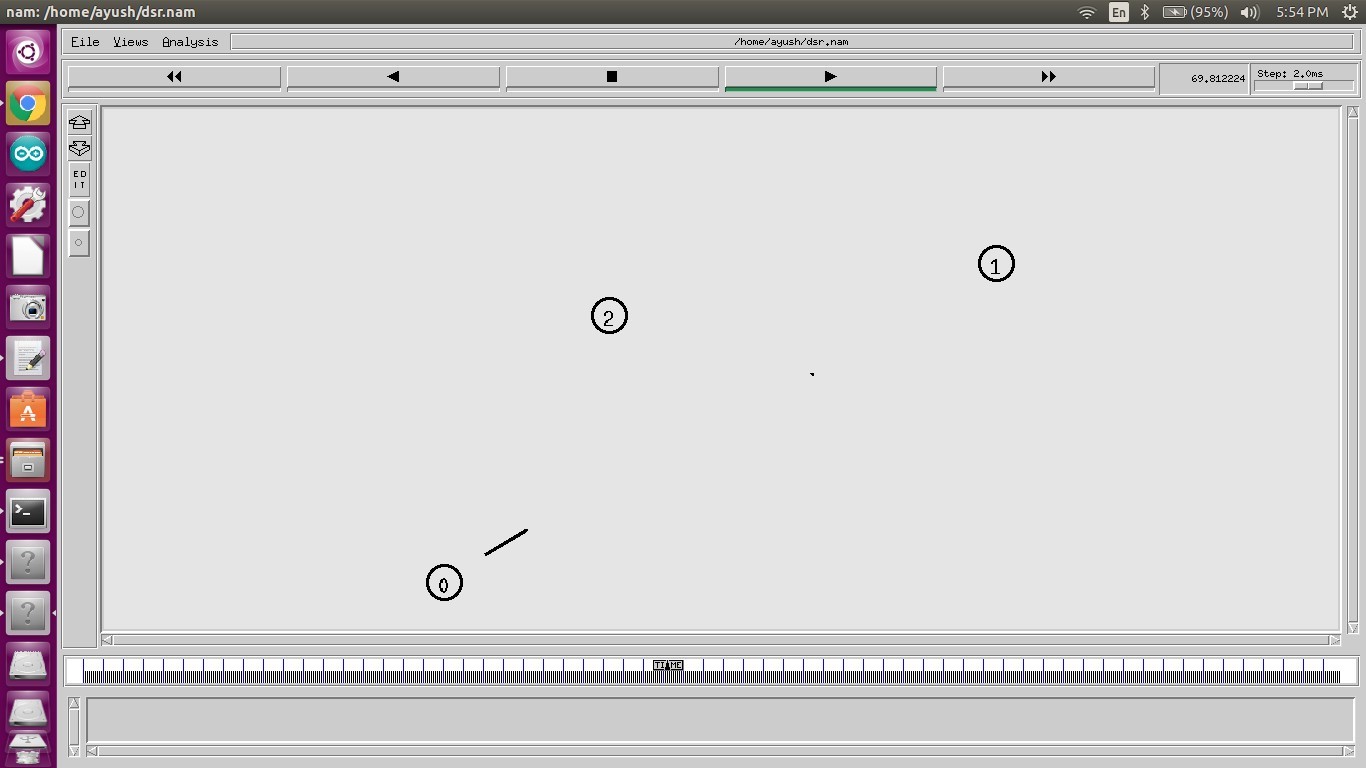


Fig 4.3 (DSR) Figure 4.1, 4.2 and 2.3 show .nam files creating a network with 22 Nodes. It depicts a network with 22 variable nodes whose behaviour has to be analyzed in the network with respect to time to determine the effecting features of each protocol. To investigate the performance of mentioned three routing protocols with varying data rates, and network load. We evaluate three parameters which shows the different nature of these Protocols, the parameters are throughput, packet success ratio and average jitter.

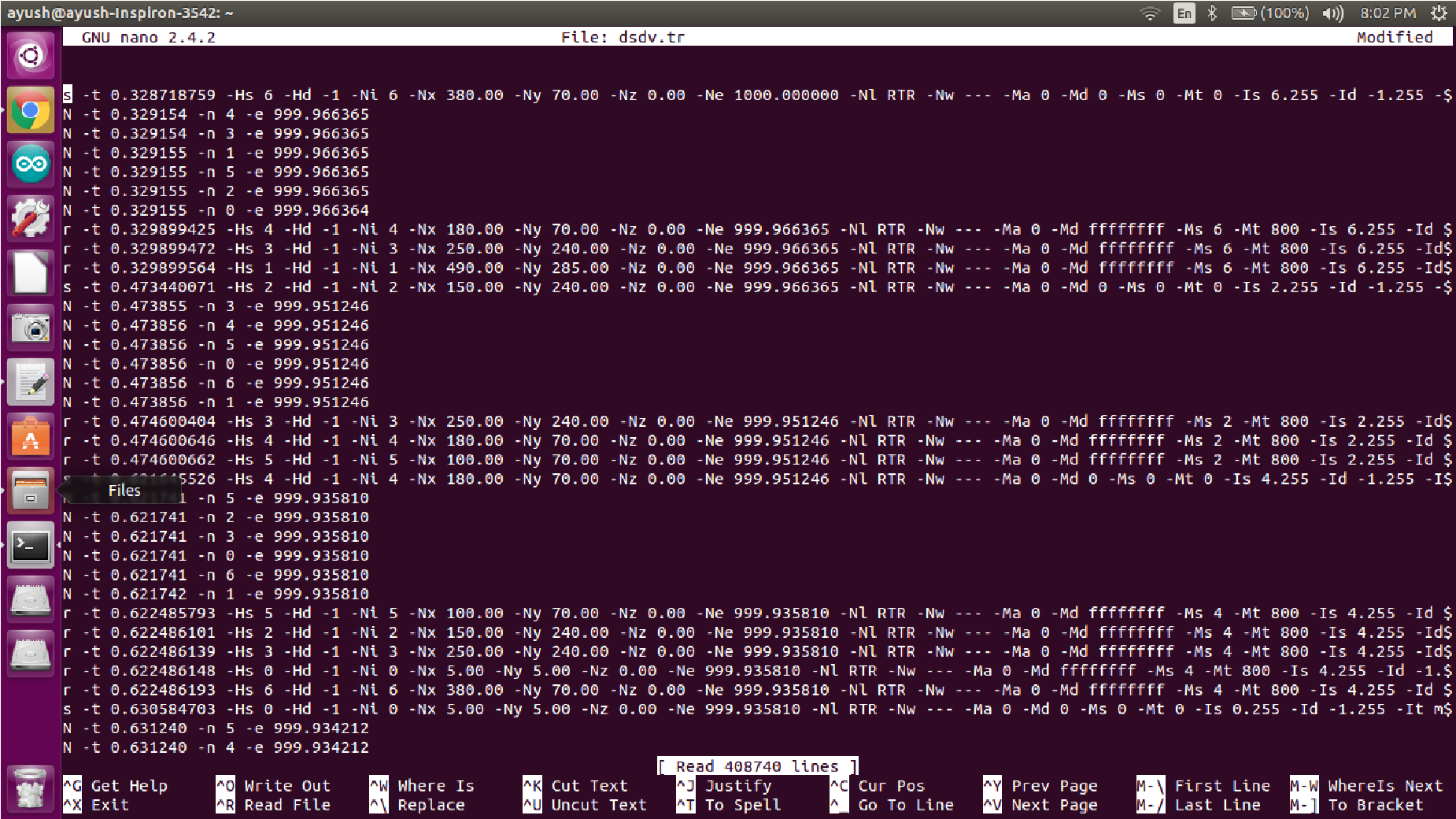


Fig 4.4 (DSDV)

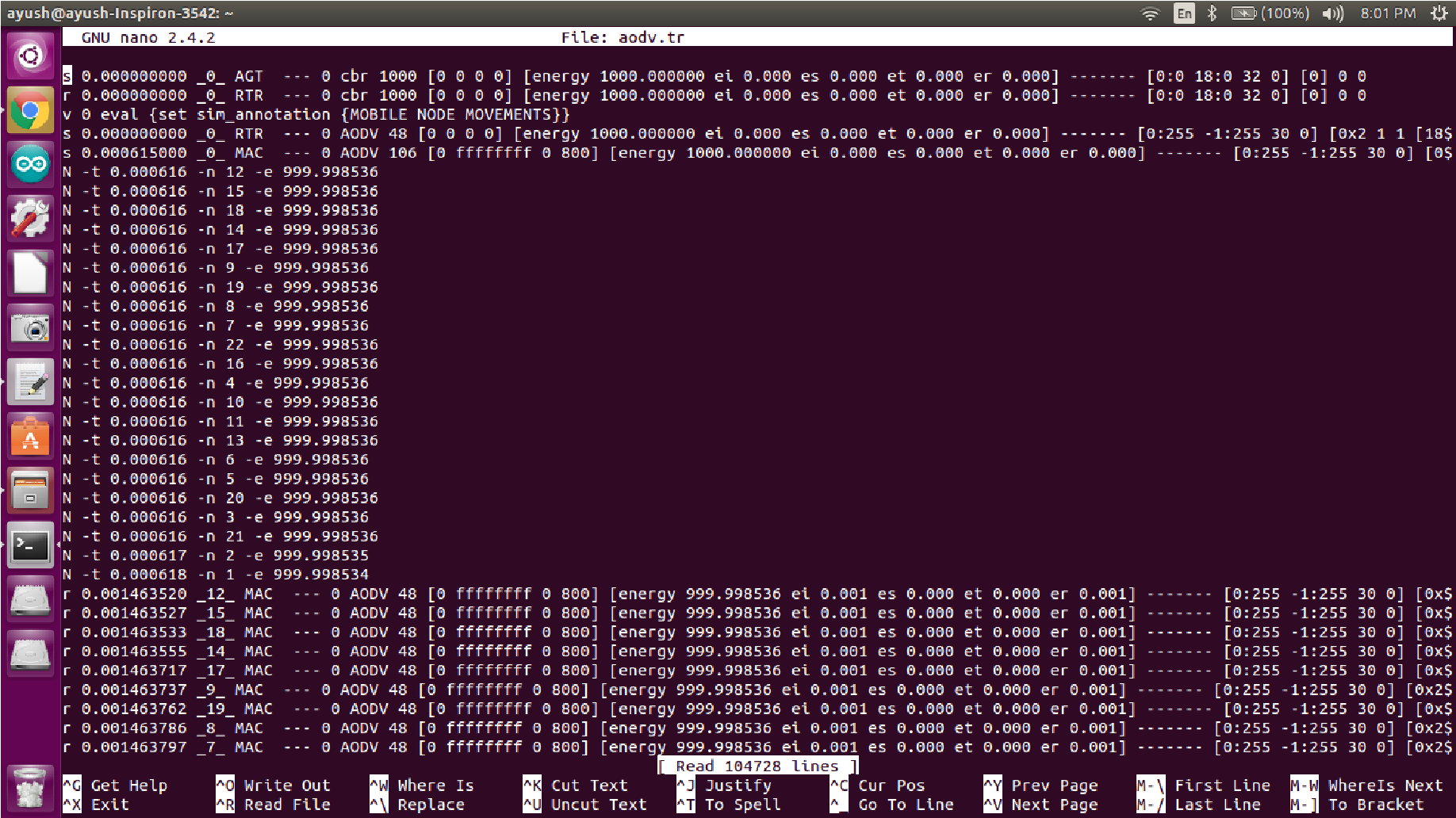


Fig 4.5 (AODV)

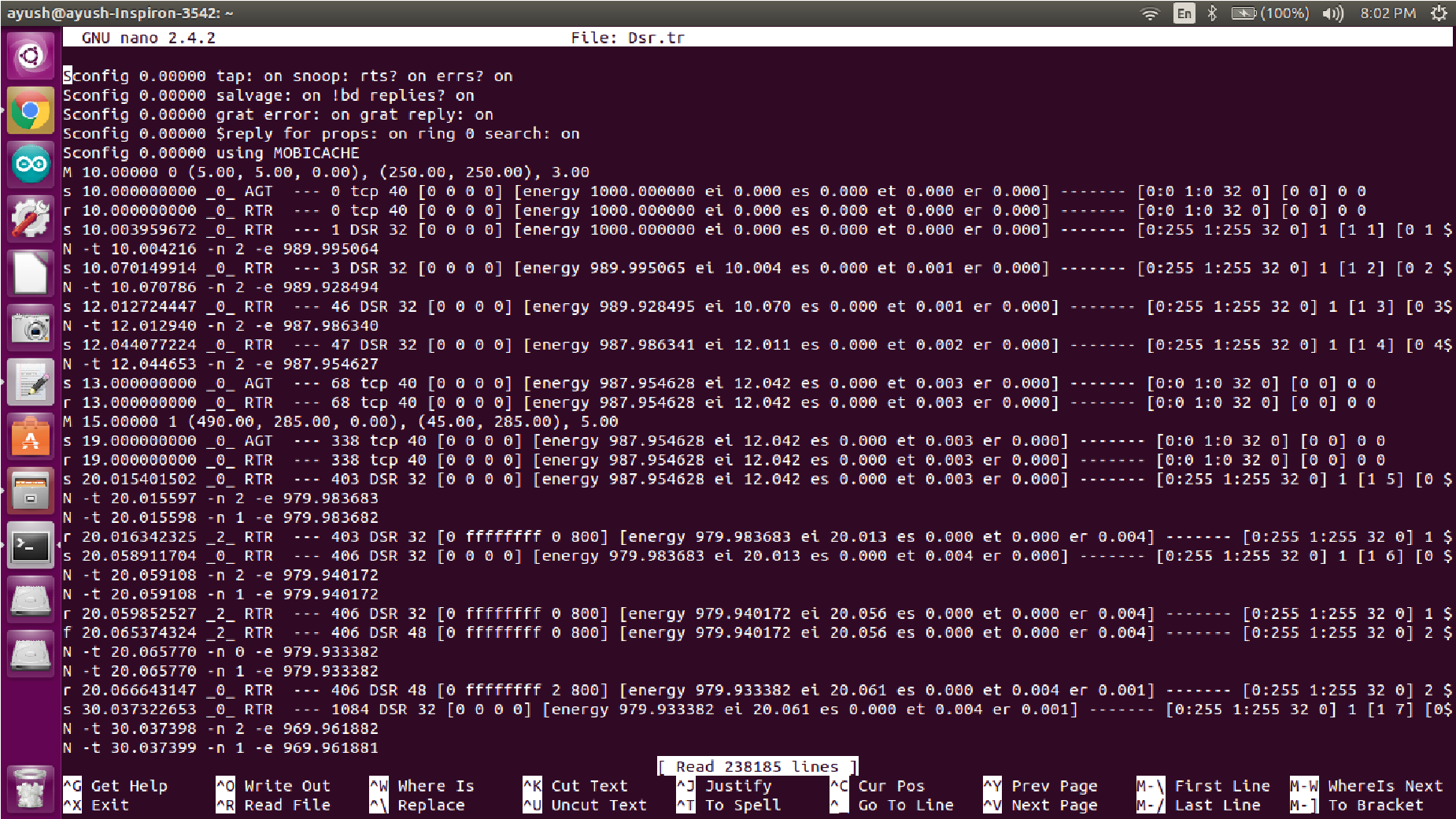


Fig 4.6(DSR)

Figure 4.4, 4.5, 4.6 show three .tr files creating trace files for all of three routing protocols taken into consideration.

1. **SIMULATOR PARAMETERS:**

We consider a network of nodes placing within a 1800m X 840m area. The performance of AODV, DSR & DSDV is evaluated by keeping the network speed and pause time constant. Table 1 shows the simulation parameters used in this valuation.

**TABLE 1**

Simulation Parameter

Simulator

Network Simulator 2

Protocol

AODV, DSDV and DSR

Simulation duration

0.58887

seconds

simulation area

1800

m \*840m

No. of nodes

AODV),3(DSDV),3(DSR

)

22(

Traffic type

CBR(AODV),TCP(DSR,DSDV)

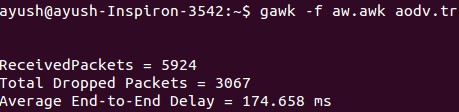
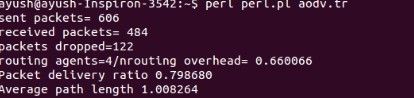
Data rate

1024

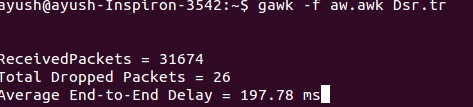
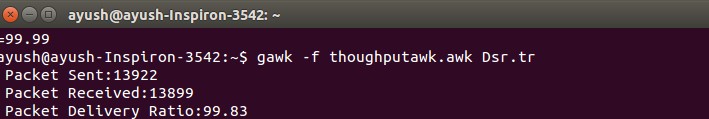
bites/sec

1. **PERFORMANCE COMPARISON**

 **Throughput, end-end delay and jitter for AODV protocol-**



##  Throughput, end-end delay and jitter for DSR protocol-



 **Throughput, end-end delay and jitter for DSDV protocol-**





From the above screenshots we can see the throughput, end-end delay, packet delivery ratio and jitter for above ad-hoc routing protocols. These parameters are important aspects for performance measurement of the network as well as the routing protocol.

We can that packet delivery ratio for DSR is more than AODV protocol but end-end delay is less than AODV protocol.

## 7. CONCLUSION

In this project, we discuss the classification of routing protocol and performance metric. We evaluate the AODV, DSR and DSDV protocol based on throughput, Packet delivery ratio, Average end to end delay and control overhead performance metrics. We see AODV is better than other these routing protocol in terms of throughput and end-to-end delay. While DSDV is better in terms of average end to end delay and DSR is better than these protocol in terms of control overhead. Overall analysis of these three routing protocol we can say that AODV is best routing protocol in MANET as compare to DSDV and DSR.

# REFERENCES

1. David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. Technical report,Carnegie Mellon University, 1996.
2. Kapang lego, pranav kumar singh and dipankar sutradhar. Comparative Study of Adhoc Routing Protocol AODV,DSR and DSDV in Mobile Adhoc NETwork, Indian Journal of Computer Science and Engineering, Vol. 1 No. 4 364-371.
3. Xiaoyan Hong, Kaixin Xu, and Mario Gerla. Scalable routing protocols for mobile ad hoc networks. 2002.
4. “Wireless Ad Hoc Networks” Zygmunt J. Haas, Jing Deng, Ben Liang, Panagiotis Papadimitratos, and S. Sajama Cornell University School of Electrical and Computer Engineering.
5. Z. J. Haas and M. R. Pearlman, “The zone routing protocol (ZRP) for ad hoc networks (InternetDraft),” Aug. 1998.
6. L. Kleinrock and K. Stevens, “Fisheye: A Lenslike Computer Display Transformation,” Technical report, UCLA, Computer Science Department, 1971.
7. Mbarushimana C, Shahrabi A, “Comparative Study of Reactive and Proactive Routing Protocols Performance in Mobile Ad Hoc Networks,” Advanced Information Networking and Applications Workshops (AINAW), 2007: P679-684.