SRI INDU COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

HANDS ON TRAINING COURSE

ON

HANDS ON TRAINING IN NS2

Date: From 25-11-2021 to 31-12-21 (6 Week Course, Only on Saturdays)

COURSE CONTENTS

MODULE -1		
Durations	Topics	Resource Person
Week 1	Introduction to NS2	Dr. C.Kotteeswaran
	Overview	
	Why TCL, Installation of NS-2	
	Network Component	
	Node and Routing	
	Packet Flow	
	Assignment-1	
Week 2	Overview of ns-2 simulation test bed	Dr. C.Kotteeswaran
	ns architecture	
	NS programming	
	TCL interpreter, characteristics, X Graph	
	Assignment-2	
Week 3	Basic Linux and Ns2	Dr. C.Kotteeswaran
	Node Commands	
	WIRELESS NETWORK PROGRAMS	
	Assignment-3	
MODULE -2		
Durations	Topics	Resource Person
	WSN program	
	Creation of TCP	
	AODV routing protocol	
	Multicast	

Week 4	Link	Dr. C.Kotteeswaran
	Assignment-4	
MODULE -3		
Durations	Topics	Resource Person
Week 5	Channel – Wireless Channel	Dr. C.Kotteeswaran
	Propagation Two Ray Ground Propagation	
	Queue Type – Drop Tail	
	Assessment -1	
	Conclusion	

1 NS 2 INTRODUCTION

Network Simulation (version 2) is one of the object-oriented language based discrete event-driven introduced at UC Berkely developed in two languages, namely C++ and OTcl

(Object Tool Command Language). Network Simulation is first and foremost used in the simulation of LAN and WAN network.

1.1 Overview

Network Simulation 2 is an event-driven simulator that simulates various kinds of IP networks. It implements network protocols such as Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), behavior of traffic source such as File Transfer Protocol (FTP), Telnet, Web, Constant Bit Rate (CBR) and Variable Bit Rate (VBR), queue management methods such as Drop Tail, RED and CBQ, and some of the routing algorithm are used. NS also works with multicasting network oriented programs and some of the Medium Access Control (MAC) layer protocols for local area network simulations. Network Simulation project is currently working for the VINT project that introduce tools for simulation results display, analysis and converters that convert network topologies generated by well-known generators to NS formats. At present, Network Simulation (version 2) developed in C++ and OTcl is on hand. This manual discusses briefly about the basic construction of NS, and explains detaily how to make use of NS frequently by giving examples.

As shown in Figure (Simplified user view of NS2), in a simplified user's view, Network simulation is an Object-oriented tool script interpretered with simulation event scheduler and the libraries of network component object, and the libraries of network setup (plumbing) module (in fact it is the plumbing modules which are implemented as member functions of the base simulator object). Otherwise to use NS, we have to program in OTcl script language.

To create and run a simulation, OTcl script should be written by the user that creates an event, initiate the network topology set up using the objects of the network and comment the traffic sources and fix the transmission time and stop time of transmitting packets through the event scheduler.

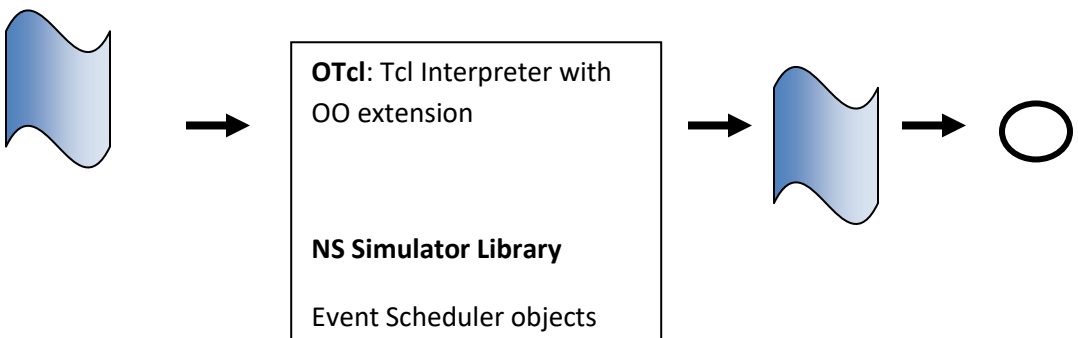


Figure Simplified User's View of NS

One more imporant component of NS beside network objects is the event scheduler. An event in NS is a packet ID that is unique for a packet with scheduled time and the pointer to an object that handles the event. In NS, an event scheduler continuously tracking of simulation time period and fires all the simulation events in the event queue programmed for the present time by invoking suitable network components, which more often than not are the ones who issued the events in the simulation, and let them perform the suitable action connected with packet pointed by the event.

Network components communicates with one another transitory packet, however this does not devour real simulation time. Each and every network components that require spending a little simulation time for handling a packet (i.e. essential delay) use the event scheduler by providing an event for the packet and to come for the event to be fired to itself previous to doing additional action handling the packet. For example, a network switch component that handles the simulation which switch with 20 microseconds of switching delay issues an event for a data packet to be switched to the scheduler as an event 20 microsecond afterward. The scheduler following 20 microseconds handle the process of dequeue the event and fires it to the switch component, which subsequently send the packet to a suitable output link component.

One more work of an event scheduler is timer. For example, Transmission Control Protocol (TCP) needs to make use of a timer to keep tracking transmission time of a packet out for further transmission (transmission of a packet with similar TCP packet number but dissimilar network packet identication). Timers will make use of the event schedulers in a same manner that delay does. The one and only difference in that timer is, it measures a time linked with a

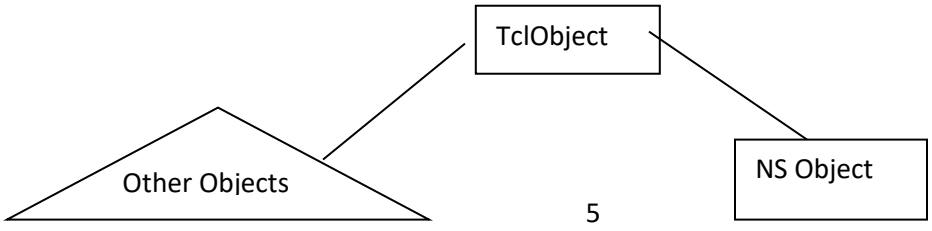
packet and does a suitable action connected to that packet after a firm time goes by, and does not simulate a time delay.

1.2 Why TCL

A user inscribe an OTcl script that's creates an event scheduler, sets up the topology of the network by make use of an objects of the network and the libraries plumbing functions, and control the traffic sources when to initiate and finalize the transmission of the packets through the event scheduler. Here plumbing is defined as a network setup, for the reason that setting up a network is plumbing possible paths for data transfer between network objects by locating the "neighbor" pointer of an object to the address of a suitable object. When a user needs to create a new network object, he or she with no trouble can make an object either by creating a fresh object or by constructing a compound object from the object library, and plumb the path of the data through the object. This may resonance like difficult job, but the plumbing OTcl modules in reality make the job trouble-free. The influence of NS comes from this plumbing.

NS is created not only in tool command language but in C++ also. For efficiency reason, NS data path implementations are separated from control path implementations. Most importantly the packet and event processing time has to be reduced, for that purpose the C++ language is used to write and compile the event scheduler and the component objects. These compiled objects are ready accessible to the OTcl interpreter through an OTcl linkage that creates a corresponding OTcl object for each and every C++ objects and makes the control functions and the configurable variables specified by the C++ object take action as member functions and member variables of the corresponding OTcl object. By the way, the controls of the C++ objects are prearranged to OTcl. It is also probable to insert member functions and variables to a C++ linked OTcl object.

1.2.1 Network Component



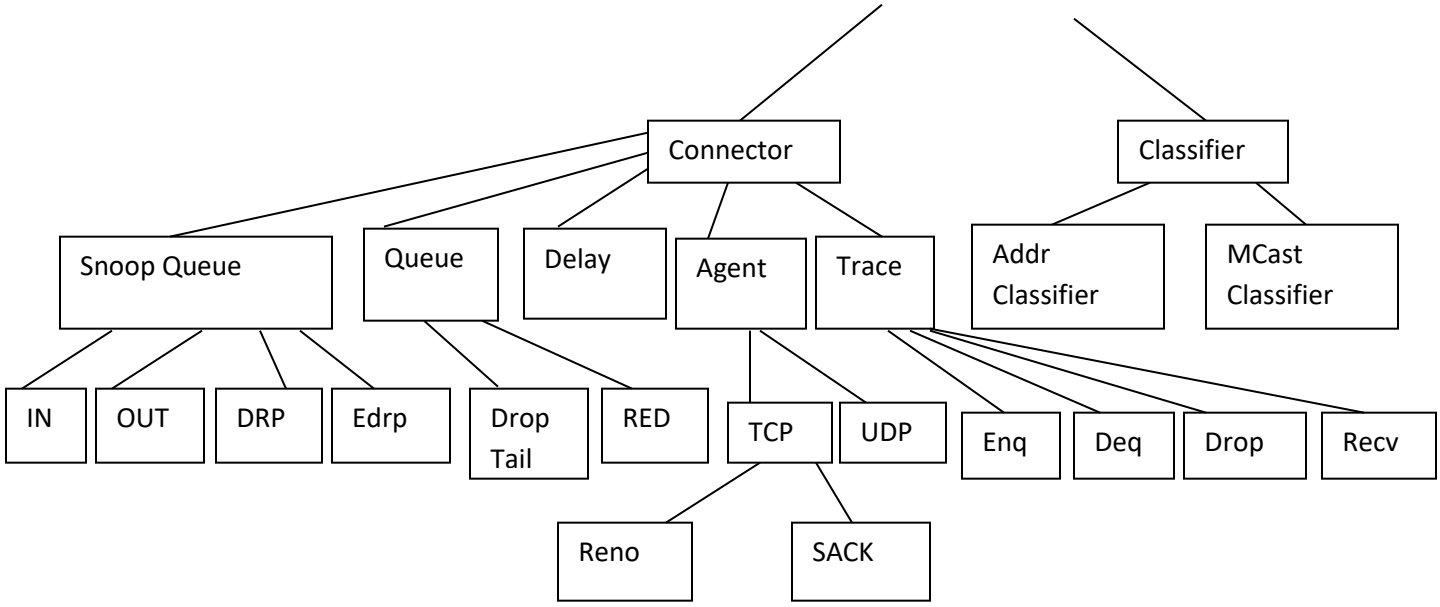


Figure Class Hierarchies (Partial)

1.2.2 Node and Routing

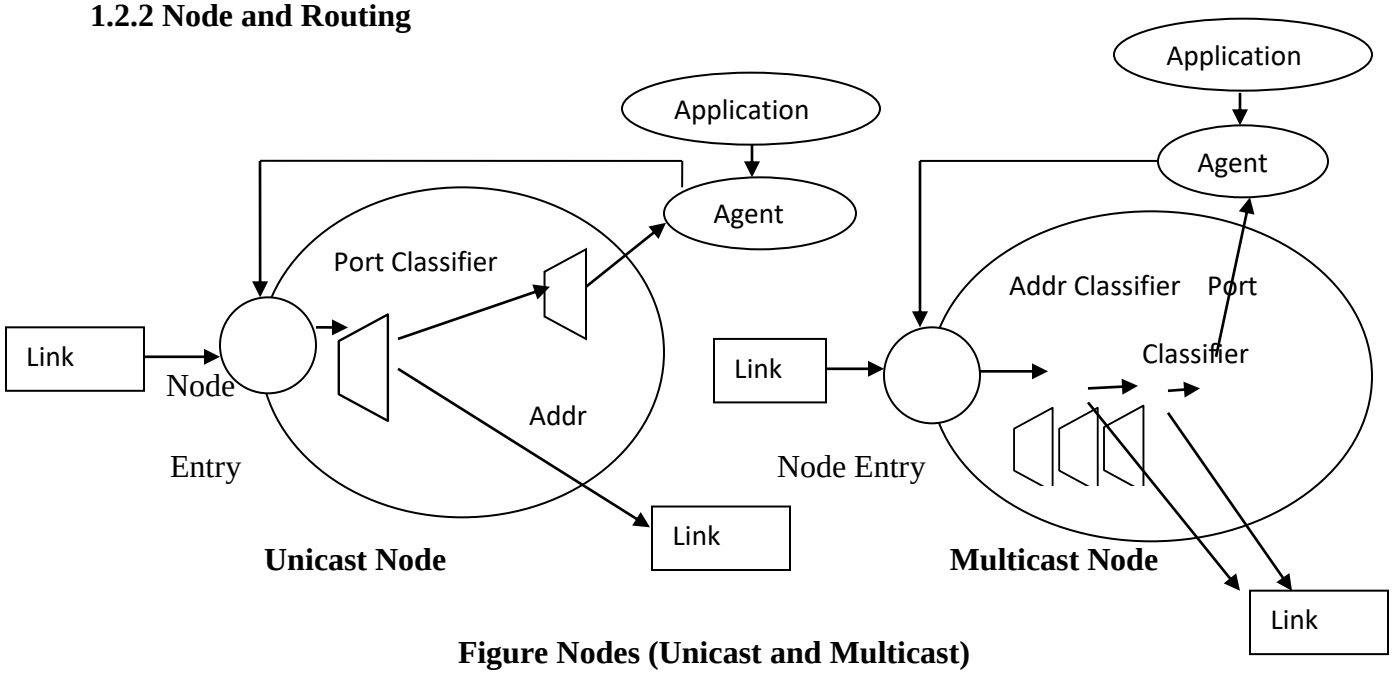


Figure Nodes (Unicast and Multicast)

A node of a network is one of a compound object composed by both the node entry object and classifiers as shown in Figure (Unicast and Multicast). There are two important types of nodes in network simulation. First is unicast node, it has an address classifier with it that does the operation of unicast routing and a port classifier. Next is multicast node, in addition to routing

and port classification, it has a classifier which classify multicast packets from unicast packets and a multicast classifier that performs multicast routing.

In network simulation, Unicast nodes are in default condition. In order to create Multicast nodes once the user must clearly notify in the input of the OTcl script, after the exact creation of scheduler object, the nodes which are created that will be in the form of multicast nodes. Next process is specification, once it get done then specify the node type, the user can also select a exact routing protocol other than using a predifined one.

Unicast

- *$ns* rtproto *type*
- *type*: Static, Session, DV, cost, multi-path

Multicast

- *$ns* multicast (right after set *$ns* [new Scheduler])

1.2.3 Link

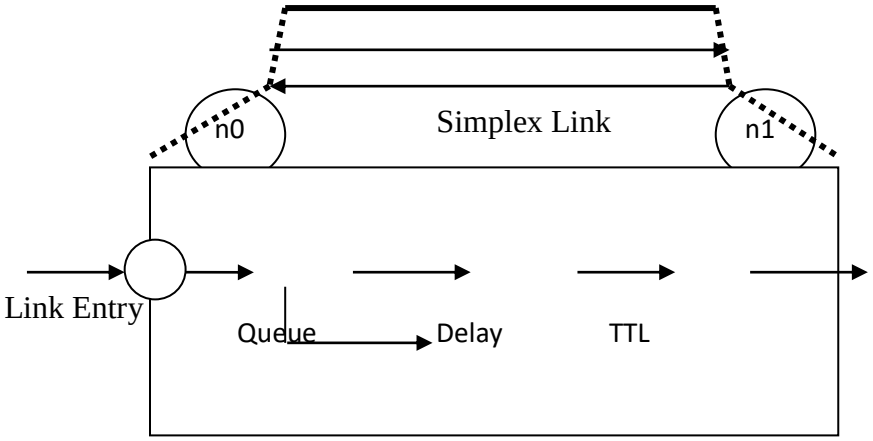


Figure Link

First notify that a node's output queue is actually implemented in the form of simplex link object. Once after completing the process of dequeued packets from a queue are passed to the Delay object that again simulates the link delay, and the dropped packets at a queue are transferred to a Null Agent and are made freed there. Finally, the TTL object calculates Time To Live (TTL) parameters for each received packets and updates.

1.2.4 Tracing

In NS2, activities of the network are traced around simplex links. If the simulator is intended for to the trace network activities (specifically make use of *$ns* trace-all *file* or *$ns* namtrace-all *file*), the links created after this commands will be followed by the upcoming trace objects inserted as shown in Figure (Inserting Trace Objects). Users creates a trace object of type *type* between the given *source* and *destination* nodes using the create-trace {*type file src dst*} command.

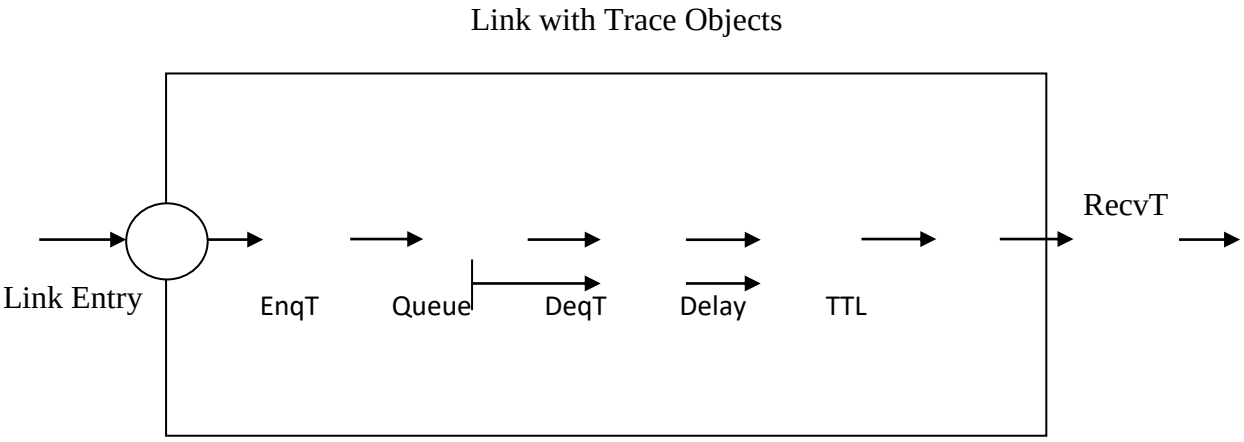


Figure Inserting Trace Objects

1.2.5 Queue Monitor

On the whole, tracing objects are intended to trace packet arrival time at which the localization is done. Even though a user gets sufficient information from the trace files, he or she might be concerned with what is going inside the output queue. For example, a user paying attention in RED queue behavior may want to calculate the dynamics of average and current queue size of a exact RED queue (i.e. need for queue monitoring). Queue monitoring can be successfully achieved using queue monitor objects and snoop queue objects as shown in Figure (Monitoring Queues).

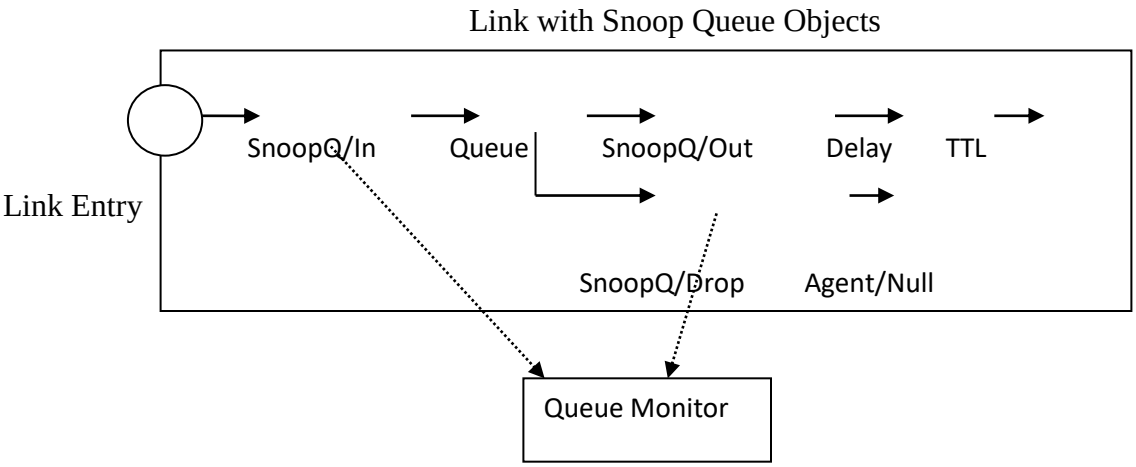


Figure Monitoring Queues

When a data packet arrives, the queue monitor object is notified by the snoop queue object of this event. Using this information the queue is monitored by the queue monitor. RED Queue Monitor Example section shows some examples for RED queue monitoring. Note that snoop queue objects can be second-hand in parallel with tracing objects although it is not shown in the above figure (Figure Monitoring Queues).

1.2.6 Packet Flow Example

Until now, the examination of two most important network components (node and link) is done. Figure (Packet Flow Examples) shows internals of an example simulation network setup and packet flow. The number of nodes network is two which is node 0 (n0) and node 1 (n1) of which the network addresses are 0 and 1 respectively. A TCP agent (sender agent) attached to n0 using port 0 communicates with a TCP sink object (destination agent) attached to n1 port 0. Finally, an FTP application layer (or traffic source) is attached to the TCP agent (sender application), asking to send some amount of data to the destination which is node 1.

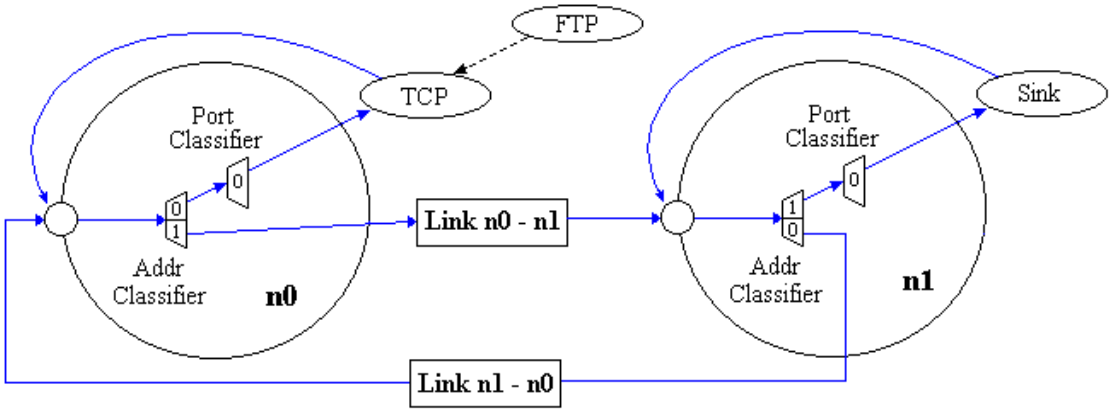


Figure Packet Flow Examples

1.3 Overview of ns-2 simulation test bed

NS-2 has many and expanding uses included.

- The performance of existing network protocols is evaluated.
- Before the usage the new network protocols are evaluated.
- To run large scale experiments and it is not possible in the real time environment.
- Various kinds of internet protocols IP are possible to simulate in network simulation 2.

NS-2 is an object oriented discrete event simulator which works to calculate the performance and behavior of the network. Simulator maintains list of events in the queue and executes one event after another event.

Features:

- Protocols mostly used
- Fast to run, with high network control
- Front end is OTCL – Object tool command language
- BACK end is C++ - Creating scenarios, extensions to C++ protocols
- fast to create and modify

1.3.1 Characteristics of NS-2

- NS-2 implementation consist of the following features
- Multicasting is employed here.

- Simulation of various kinds of wireless networks
- Terrestrial (cellular, Adhoc, GPRS, WLAN, BLUETOOTH), satellite network are used
- IEEE 802.11 standard can be simulated, Mobile Internet Protocols and Ad hoc protocols such as DSR, TORA, DSDV and AODV Routing are simulated

1.3.2 Software Tools used with NS-2

In the simulation, there are the two tools are used.

- **NAM(Network Animator)**
- **xGraph**

1.3.3 NS ARCHITECTURE

- Object-oriented (C++, OTCL).
- Modular approach
- Fine –grained object composition
- Reusability
- Maintenance
- Perfomanace(speed and memory)
- Careful planning of modularity

1.3.4 NS PROGRAMMING

- Create the event scheduler
- Turn on tracing
- Create network
- Setup routing
- Insert errors
- Create transport connection
- Create traffic
- Transmit application-level data

1.3.5 TCL INTERPRETER

TclCL is the language used to provide a linkage between C++ and OTcl. Toolkit Command Language(Tcl/OTcl) scripts are written to set up/configure network topologies. TclCL provides linkage for class hierarchy,object instantiation, variable binding and command dispatching. OTcl is used for periodic or triggered events

The following is written and compiled with C++

1. Events Scheduler
2. NAM- The Network Animator
3. Xgraph- For plotting
4. Pre Processing- Traffic & Topology generator
5. Post Processing- Simple Trace Analysis often used TCL and Pearl

1.3.6 CHARACTERISTICS

NS-2 implements the following features

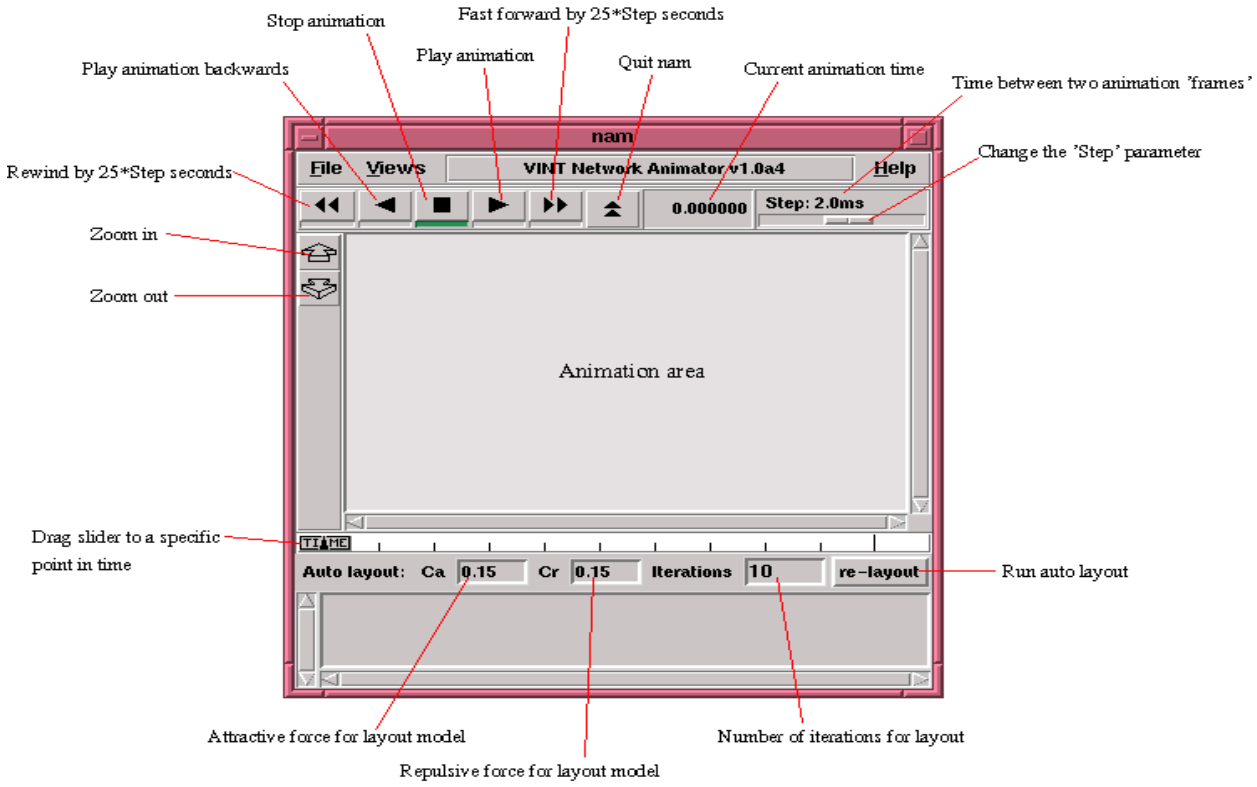
1. Router queue Management Techniques Drop Tail, RED, CBQ,
2. Multicasting
3. Simulation of wireless networks
4. Developed by Sun Microsystem + UC Berkeley (Daedalus project)
5. Terrestrial (Cellular, Ad-hoc, GPRS, WLAN, BLUETOOTH), Satellite

1.3.7 NAM (Network Animator)

NAM provides a visual interpretation of the network topology created. The application was developed as part of the VINT project. Its feature is as follows.

- Provides a visual interpretation of the network created
- Can be executed directly from a Tcl script

- Controls include play; stop fast forward, rewind, pause, a display speed controller button and a packet monitor facility.
- Presented information such as throughput, number packets on each link



1.3.8 X Graph

X- Graph is an X-Window application that includes:

Interactive plotting and graphing Animated and derivatives, to use Graph in NS-2 the executable can be called within a TCL script. This will then load a graph displaying the information visually displaying the information of the file produced from the simulation. The output is a graph of size 800 x 400 displaying information on the traffic flow and time.

1.3.9 Simulation tool

NS2 are often growing to include new protocols. LANs need to be updated for new wired/wireless support. ns are an object oriented simulator, written in C++, with an OTcl interpreter as a front-end. The simulator supports a class hierarchy in C++ and a similar class hierarchy within the OTcl interpreter (also called the interpreted hierarchy). The two hierarchies

are closely related to each other; from the user's perspective, there is a one-to-one correspondence between classes in the interpreted.

2 Basic Linux and Ns2

2.1 Linux Commands

cd : change directory

- Syntax: cd directoryname

ls : list the files in current directory

- Syntax: ls

rm : Remove a file from directory

- Syntax: rm filename

cp : Copying file from one directory to another

- Syntax: cp filename directoryname

pwd: For checking current directory

- Syntax: pwd

ps: For viewing currently running processes on system

- Syntax: ps

kill: For killing a process

- Syntax: kill processid

cat: For viewing file contents on terminal

- Syntax: cat filename

clear: clear the contents on terminal

- Syntax: clear

gcc: For compliling c and c++ programs.

- Syntax: gcc programname.c

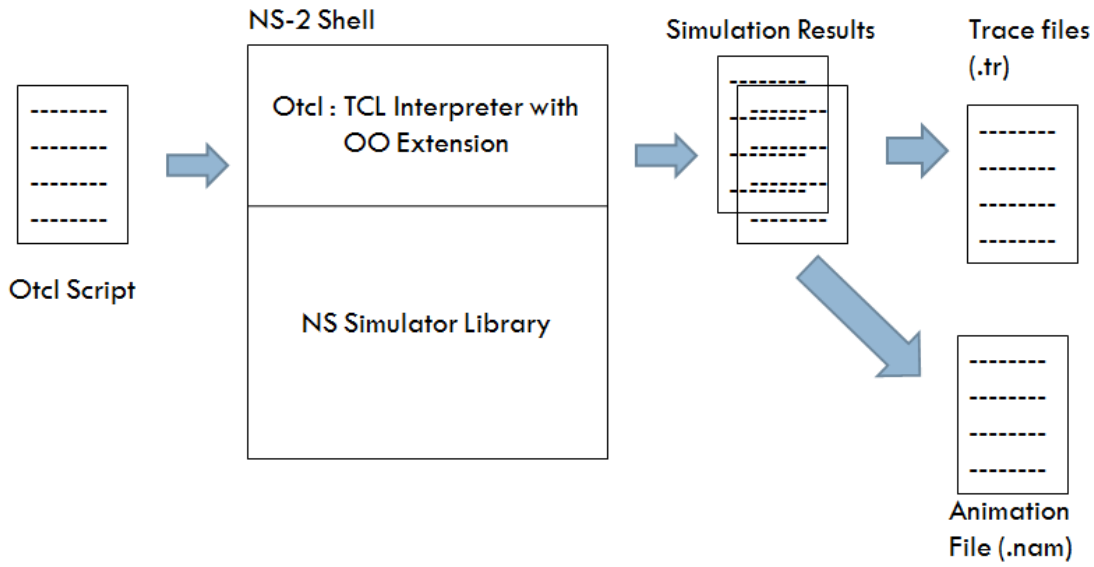
gedit: Create and open the file in text editor.

- Syntax: gedit filename

./ : For running object file.

- Syntax: ./ objectfilename

2.2 Simulation System Architecture



2.3 Installation of NS-2

2.3.1 Installation on Linux

Copy ns-allinone-2.34.tar_1.gz into /usr/local folder

Extract ns-allinone-2.34.tar_1.gz, you will get ns-allinone-2.34.tar_1.

Extract ns-allinone-2.34.tar_1, you will get ns-allinone-2.34 folder.

Go to ns-allinone-2.34 folder and say – (./install).

Go to ns-2.34 folder,

Do (./configure)

Do make all.

Do make install.

2.3.2 Bash file setting (option 2)

Open Terminal

Type on terminal following command

gedit ~/.bashrc

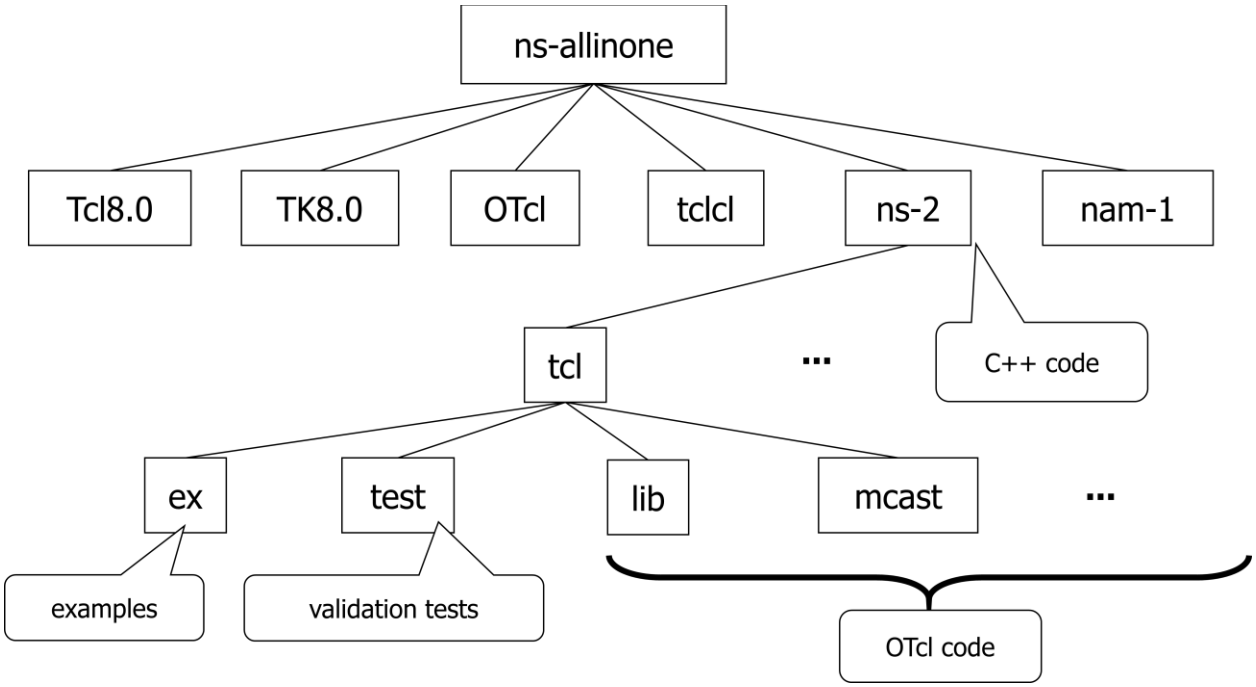
Add the TCL library, LD library and ns library path in .bashrc file.

Save the changes

Type on terminal following comand

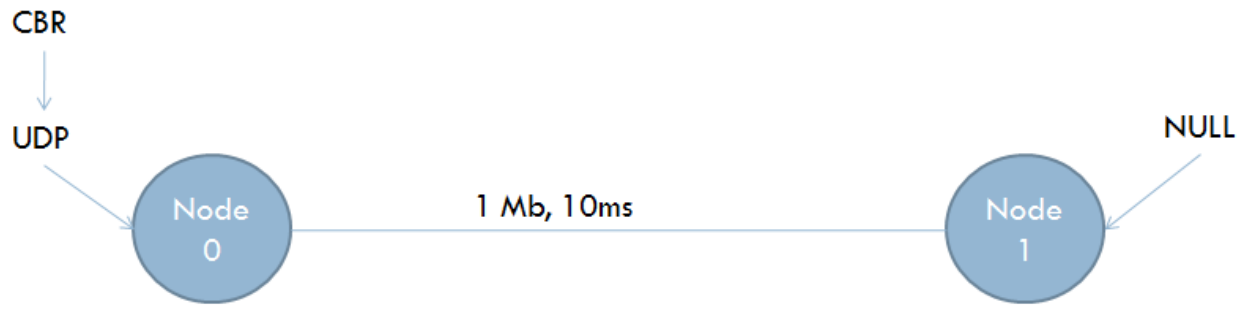
source ~/.bashrc

2.4 NS-2 Directory Structure



3 Scenarios

3.1 First Simulation Scenario



Simulation Script

```
#Create a simulator object
set ns [new Simulator]
#Open the nam trace file
set nf [open s1.nam w]
$ns namtrace-all $nf
#Open the event trace file
set nf1 [open s1.tr w]
$ns trace-all $nf1
#Define a 'finish' procedure
proc finish { }
{
    global ns nf
    $ns flush-trace
        #Close the nam trace file
    close $nf
 #Close the event trace file
    close $nf1
   #Execute nam
    exec nam s1.nam &
    exit 0
}
# Create two nodes
set n0 [$ns node]
set n1 [$ns node]
#Create a duplex link between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

```
#Create a UDP agent
set udp0 [new Agent/UDP]
#Attach udp agent to node n0
$ns attach-agent $n0 $udp0
# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
#Create a Null agent (a traffic sink) and attach it to node n1
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
#Connect the traffic source with the traffic sink
$ns connect $udp0 $null0
#Schedule events for the CBR agent
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```

Save the simulation script in specific folder.

Open the terminal and go up to specific folder.

Run the simulation script,

ns: command to run simulation script.

Syntax: ns filename.tcl

e.g. ns First_script_wired.tcl

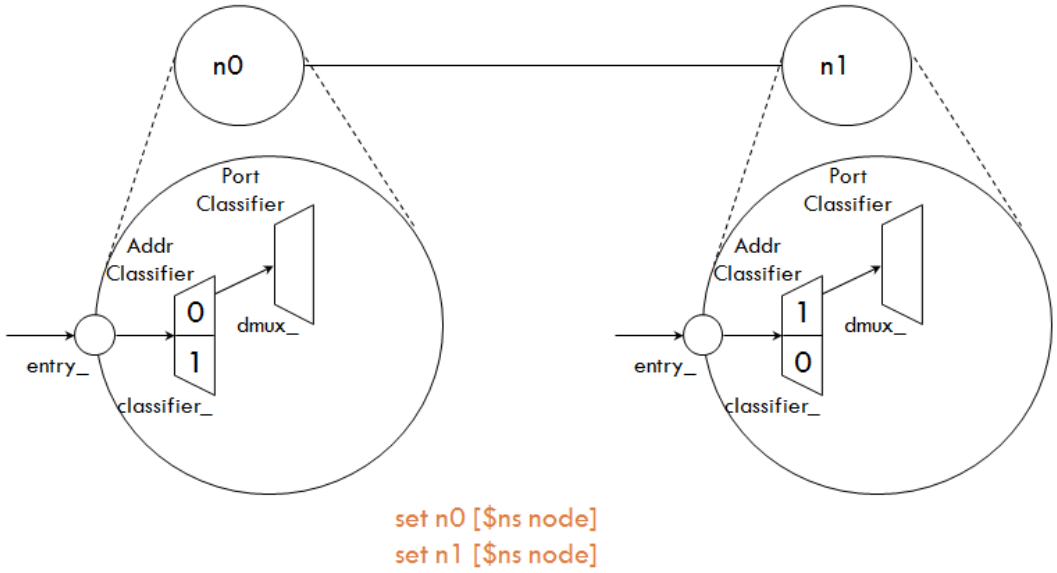
Run the nam file,

nam: command to run animation file

Syntax: nam filename.nam

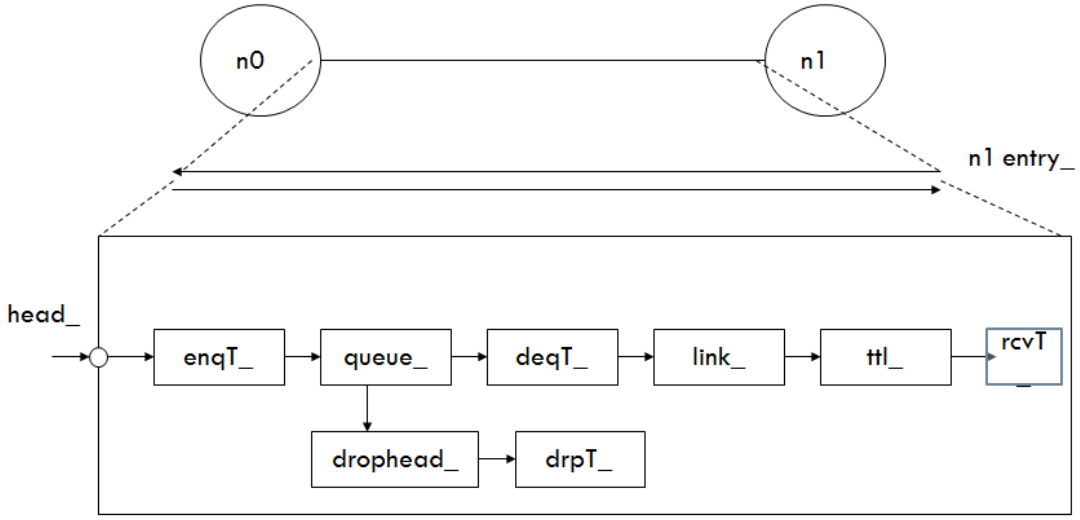
e.g. nam s1.nam

3.1.1 Flow of Simulation (NS-Node)



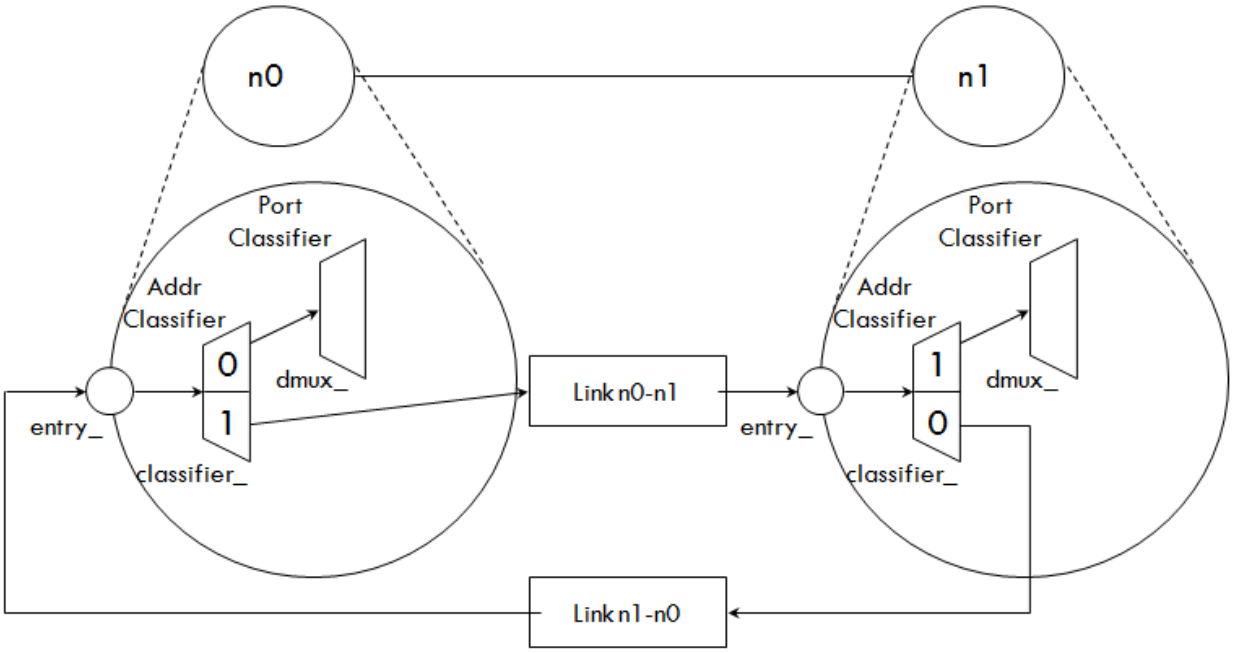
set n0 [$ns node]
set n1 [$ns node]

3.1.2 Flow of Simulation (Network Topology – Link)

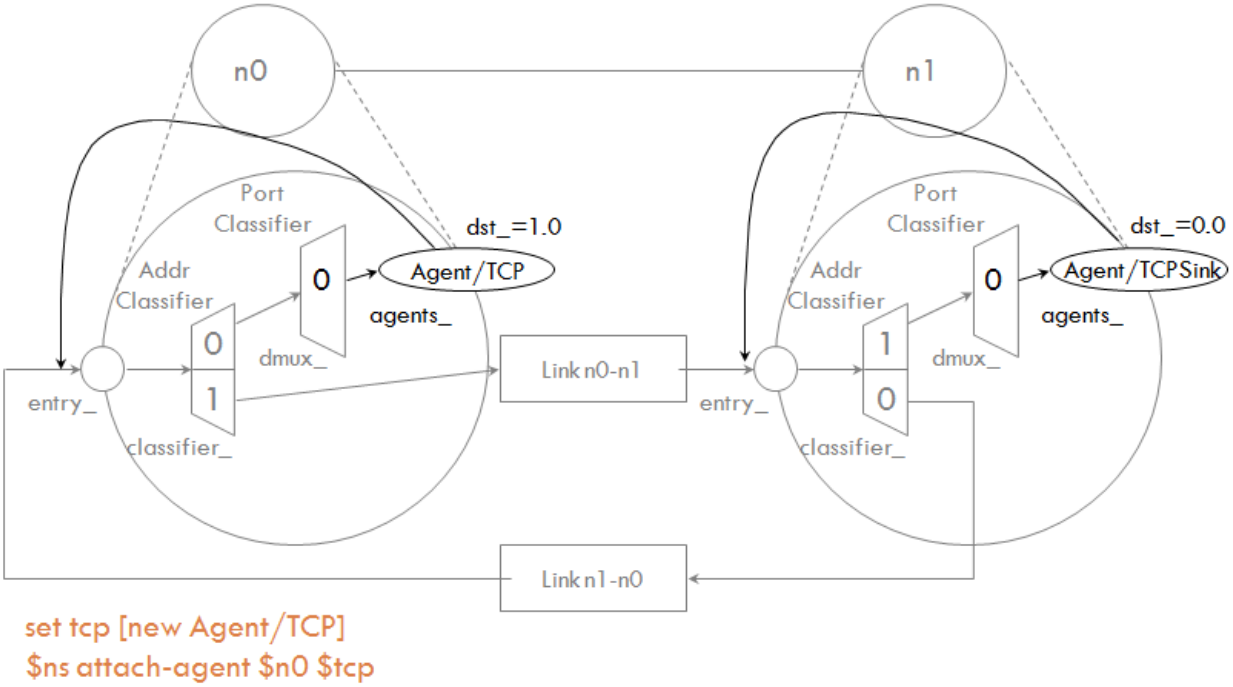


$ns duplex-link $n0 $n1 1Mb 10ms DropTail

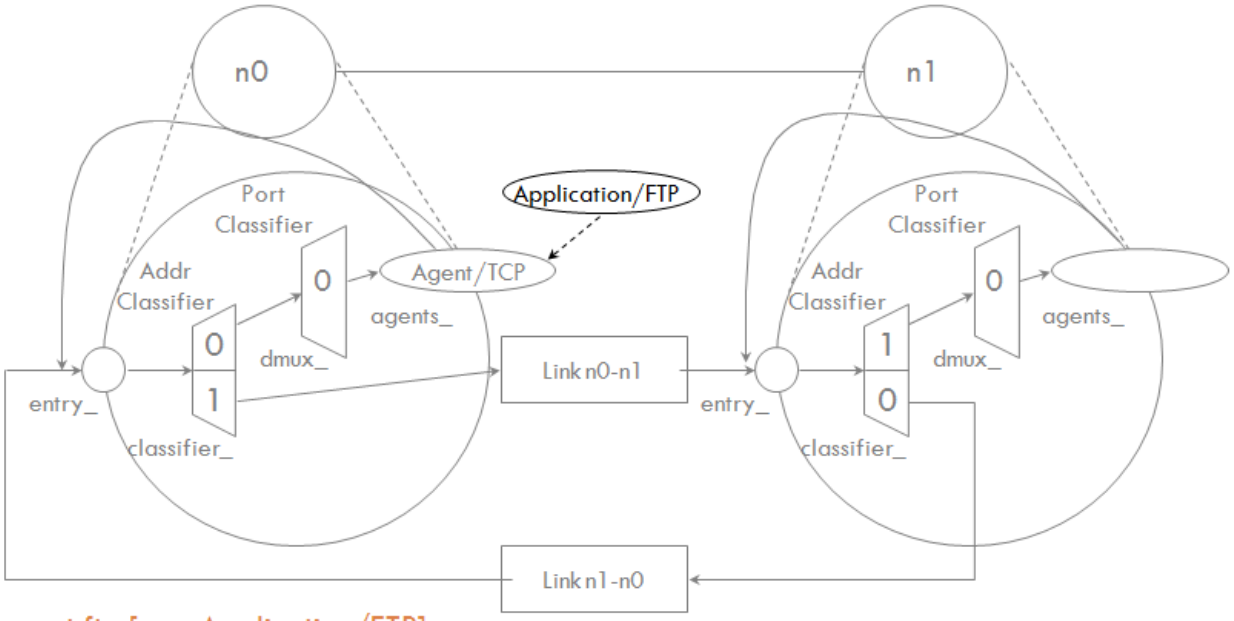
3.1.3 Flow of Simulation (Routing)



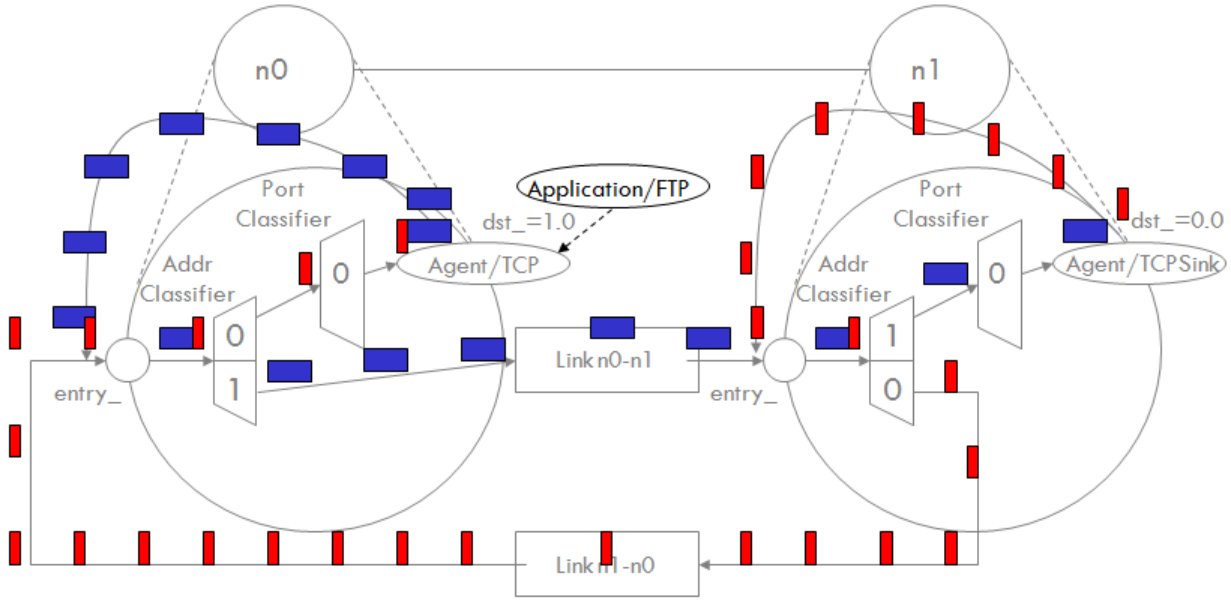
3.1.4 Flow of Simulation (Transport)



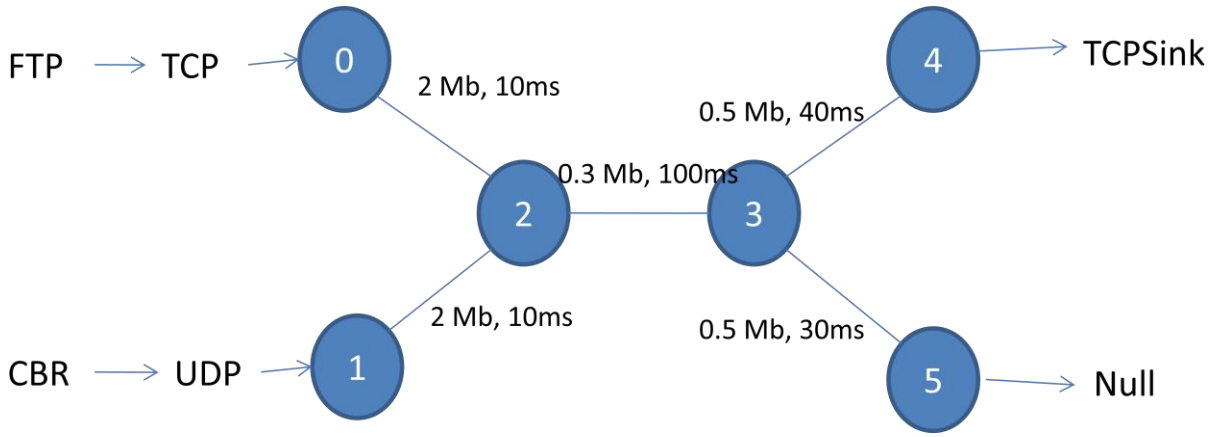
3.1.5 Flow of Simulation (Application)



3.1.6 Flow of Simulation (Packet Flow)



3.2 Second Simulation Scenario



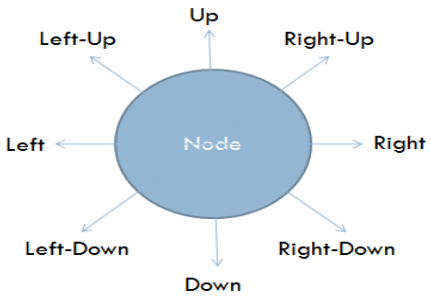
3.2.1 Simulation Script 2

```
# Create Simulator Object
    set ns [new Simulator]
#Define different colors for data flows (for NAM)
    $ns color 1 Blue
    $ns color 2 Red
#Open the Event trace files
    set file1 [open out.tr w]
    $ns trace-all $file1
#Open the NAM trace file
    set file2 [open out.nam w]
    $ns namtrace-all $file2
#Define a 'finish' procedure
    proc finish {} {
        global ns file1 file2
         $ns flush-trace
        close $file1
        close $file2
        exec nam out.nam &
        exit 0  }
#Create six nodes
    set n0 [$ns node]
    set n1 [$ns node]
    set n2 [$ns node]
    set n3 [$ns node]
    set n4 [$ns node]
```

```
    set n5 [$ns node]
#Label to the node n1 and node n2
$ns at 0.1 "$n1 label \"CBR\""
$ns at 1.0 "$n0 label \"FTP\""
#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail
$ns duplex-link $n3 $n4 0.5Mb 40ms DropTail
$ns duplex-link $n3 $n5 0.5Mb 30ms DropTail
#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns simplex-link-op $n2 $n3 orient right
$ns simplex-link-op $n3 $n2 orient left
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n3 $n5 orient right-down
#Set Queue Size of link (n2-n3) to 40
$ns queue-limit $n2 $n3 40
#Setup a TCP connection
    set tcp [new Agent/TCP]
    $ns attach-agent $n0 $tcp
    set sink [new Agent/TCPSink]
    $ns attach-agent $n4 $sink
```

```
        $ns connect $tcp $sink
        $tcp set fid_ 1
        $tcp set window_ 8000
        $tcp set packetSize_ 552
#Setup a FTP over TCP connection
        set ftp [new Application/FTP]
        $ftp attach-agent $tcp
        $ftp set type_ FTP
#Setup a UDP connection
        set udp [new Agent/UDP]
        $ns attach-agent $n1 $udp
        set null [new Agent/Null]
        $ns attach-agent $n5 $null
        $ns connect $udp $null
        $udp set fid_ 2
#Setup a CBR over UDP connection
        set cbr [new Application/Traffic/CBR]
        $cbr attach-agent $udp
        $cbr set type_ CBR
        $cbr set packet_size_ 1000
        $cbr set rate_ 0.01mb
        $cbr set random_ false
# Scheduling the event
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"


$ns at 624.0 "$ftp stop"
$ns at 624.5 "$cbr stop"
# Trace Congestion Window and RTT
set file [open cwnd_rtt tr w]
$tcp attach $file
$tcp trace cwnd_
$tcp trace rtt_
# Call finish procedure
$ns at 625.0 "finish"
# Run the simulation
$ns run
```

3.3 Node Orientation



3.4 Node Commands

$ns node [<hier_addr_>]

$ns node-config -<config-parameters> <optional-val>

$node id

$node node-addr

$node reset

$node agent <port_num>

$node entry

$node attach <agent>

$node detach <agent>

$node neighbors

$node add-neighbor <neighbor_node>

$node add-route <destination_id> <target>

$node alloc-port <null_agent>

$node incr-rgtable-size

More Node Commands

Check ~ns-2.34/tcl/lib/ns-node.tcl and ~tcl/lib/ns-mobilenode.tcl

3.5 Link Commands

$ns simplex-link <node1> <node2> <bw> <delay> <qtype> <args>

$ns duplex-link <node1> <node2> <bw> <delay> <qtype> args>

$ns simplex-link-op <n1> <n2> <op> <args>

$ns duplex-link-op <n1> <n2> <p> <args>

$ns lossmodel <lossobj> <from> <to>

$link head

$link link

$link add-to-head <connector>

$link queue

$link cost <c>

$link cost?

$link if-lable?

$link up

$link down

$link up?

$link all-connectors <op>

More Node Commands

Check ~ns-2.34/tcl/lib (ns-lib.tcl,ns-link.tcl, ns-intserv.tcl, ns-namsupp.tcl, ns-queue.tcl) and ~tcl/mcast (McastMonitor.tcl, ns-mcast.tcl), ~ns-2.34/tcl/session/session.tcl

3.6 Simulator Commands

set ns [new Simulator]

set now [$ns now]

$ns halt

$ns run

$ns at <time> <event>

$ns cancel <event>

$ns flush-trace

$ns use - scheduler <type>

$ns after <delay> <event>

$ns clearMemTrace

$ns is-started

$ns dumpq

More functions

Check ~ns-2.34/tcl/lib/ns-lib.tcl, ~ns-2.34/commnon/scheduler.{cc,h} and ~ns-2.34/heap.h

3.7 Trace Related Commands

$ns trace-all <trace-file>

$ns namtrace-all <namtracefile>

$ns namtrace-all-wireless <namtracefile> <X> <Y>

$ns nam-end-wireless <stoptime>

$ns flush-trace

$ns create-trace <type> <file> <src> <dst> <optional:op>

$ns trace-queue <n1> <n2> <optional : file>

$ns namtrace-queue <n1> <n2> <optional : file>

$ns drop-trace <n1> <n2> <trace>

$ns monitor-queue <n1> <n2> <qtrace> <optional : sampleinterval>

$link trace-dynamics <ns> <fileID>

More Functions,

Check ~ns-2.34/trace.{cc,h}, ~ns-2.34/tcl/lib/ns-trace.tcl, ~ns/queue-monitor.{cc,h}, ~ns-2.34/tcl/ns-link.tcl, ~ns-2.34/packet.h, ~ns-2.34/flowmon.cc and ~ns-2.34/classifier-hash.cc

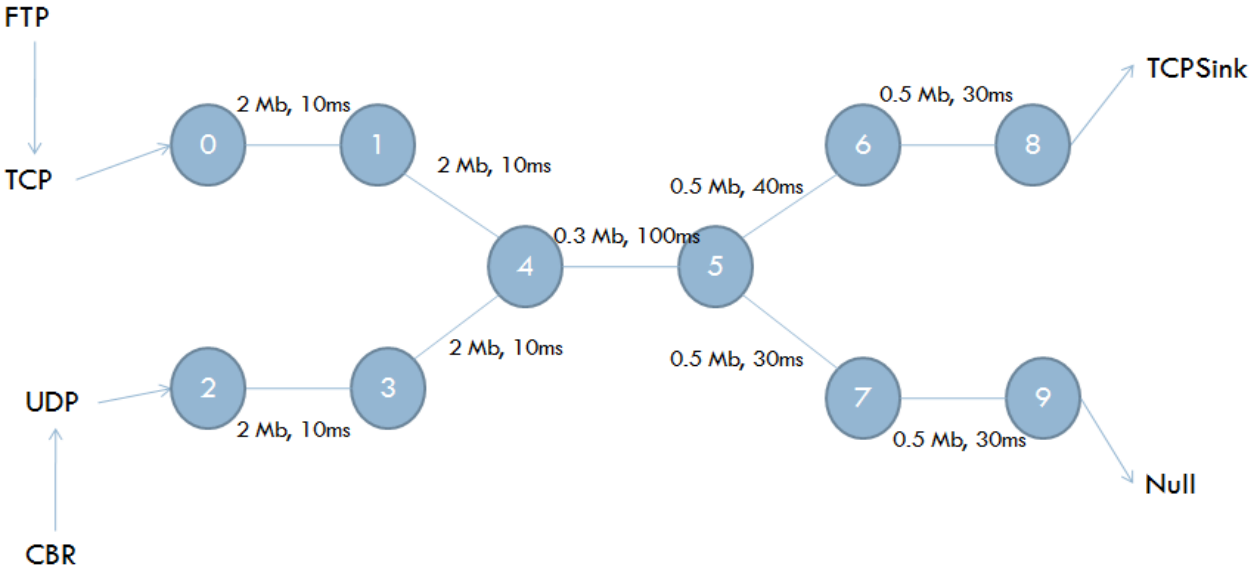
3.8 NAM Commands

$ns color <color-id>

$ns trace-annotate <annotation>

$ns set-animation-rate <timestep>

3.9 Third Simulation Scenario



3.9.1 Simulation Script 3

```
# Create Simulator Object
    set ns [new Simulator]
#Define different colours for data flows (for NAM)
    $ns color 1 Blue
    $ns color 2 Red
#Open the Event trace files
    set file1 [open out.tr w]
    $ns trace-all $file1
#Open the NAM trace file
    set file2 [open out.nam w]
    $ns namtrace-all $file2
# Define a 'finish' procedure
    proc finish {} {
        global ns file1 file2
          $ns flush-trace
        close $file1
        close $file2
        exec nam out.nam &
        exit 0 }
#Create ten nodes
    set n0 [$ns node]
    set n1 [$ns node]
    set n2 [$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]
#Label to the node n1 and node n2
$ns at 0.1 "$n1 label \"CBR\""
$ns at 1.0 "$n0 label \"FTP\""
#Create links between the nodes
$ns duplex-link $n0 $n1 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 2Mb 10ms DropTail
$ns duplex-link $n1 $n4 2Mb 10ms DropTail
$ns duplex-link $n3 $n4 2Mb 10ms DropTail
$ns simplex-link $n4 $n5 0.3Mb 100ms DropTail
$ns simplex-link $n5 $n4 0.3Mb 100ms DropTail
$ns duplex-link $n5 $n6 0.5Mb 40ms DropTail
$ns duplex-link $n6 $n8 0.5Mb 40ms DropTail
$ns duplex-link $n5 $n7 0.5Mb 30ms DropTail
$ns duplex-link $n7 $n9 0.5Mb 30ms DropTail
#Set Queue Size of link (n4-n5) to 10
$ns queue-limit $n4 $n5 10
```

```
#Setup a TCP connection
    set tcp [new Agent/TCP]
    $ns attach-agent $n0 $tcp
    set sink [new Agent/TCPSink]
    $ns attach-agent $n8 $sink
    $ns connect $tcp $sink
    $tcp set fid_ 1
    $tcp set window_ 8000
    $tcp set packetSize_ 552
#Setup a FTP over TCP connection
    set ftp [new Application/FTP]
    $ftp attach-agent $tcp
    $ftp set type_ FTP
#Setup a UDP connection
    set udp [new Agent/UDP]
    $ns attach-agent $n2 $udp
    set null [new Agent/Null]
    $ns attach-agent $n9 $null
    $ns connect $udp $null
    $udp set fid_ 2
#Setup a CBR over UDP connection
    set cbr [new Application/Traffic/CBR]
    $cbr attach-agent $udp
```

```
    $cbr set type_ CBR
    $cbr set packet_size_ 1000
    $cbr set rate_ 0.01mb
    $cbr set random_ false
# Scheduling the event
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 624.0 "$ftp stop"
$ns at 624.5 "$cbr stop"
# Trace Congestion Window and RTT
set file [open cwnd_rtt.tr w]
$tcp attach $file
$tcp trace cwnd_
$tcp trace rtt_
# Call finish procedure
    $ns at 625.0 "finish"
# Run the simulation
    $ns run
```

3.10 Wired file format

event	time	from node	to node	pkt type	pkt size	flags	fid	src addr	dst addr	seq num	pkt id

```
r : receive (at to_node)
+ : enqueue (at queue)
- : dequeue (at queue)
d : drop    (at queue)

src_addr : node.port (3.0)
dst_addr : node.port (0.0)
```

Event	Time	From-Node	To-Node	Pkt-Type	Pkt-Size	Flags	Fid	Src-addr	Dest-addr	Seq-num	Pkt-id
-	1.06	0	2	tcp	1040	-------	1	0.0	3.0	2	124
r	1.07	1	2	cbr	1000	-------	2	1.0	3.1	120	122
+	1.07	2	3	cbr	1000	-------	2	1.0	3.1	120	122
d	1.07	2	3	cbr	1000	-------	2	1.0	3.1	120	122

4 WIRELESS NETWORK PROGRAMS

4.1 SIMULATION PROGRAM FOR LAN NETWORK

```
set ns [new Simulator]

#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the Trace files
set file1 [open out.tr w]
set winfile [open WinFile w]
$ns trace-all $file1

#Open the NAM trace file
set file2 [open out.nam w]
$ns namtrace-all $file2

#Define a 'finish' procedure
proc finish {} {
        global ns file1 file2
        $ns flush-trace
        close $file1
        close $file2
        exec nam out.nam &
        exit 0
}

#Create six nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$n1 color red
$n1 shape box
```

#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail

set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail
MAC/Csma/Cd Channel]

$ns duplex-link $n3 $n4 0.5Mb 40ms DropTail
$ns duplex-link $n3 $n5 0.5Mb 30ms DropTail

#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns simplex-link-op $n2 $n3 orient right
$ns simplex-link-op $n3 $n2 orient left
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n3 $n5 orient right-down

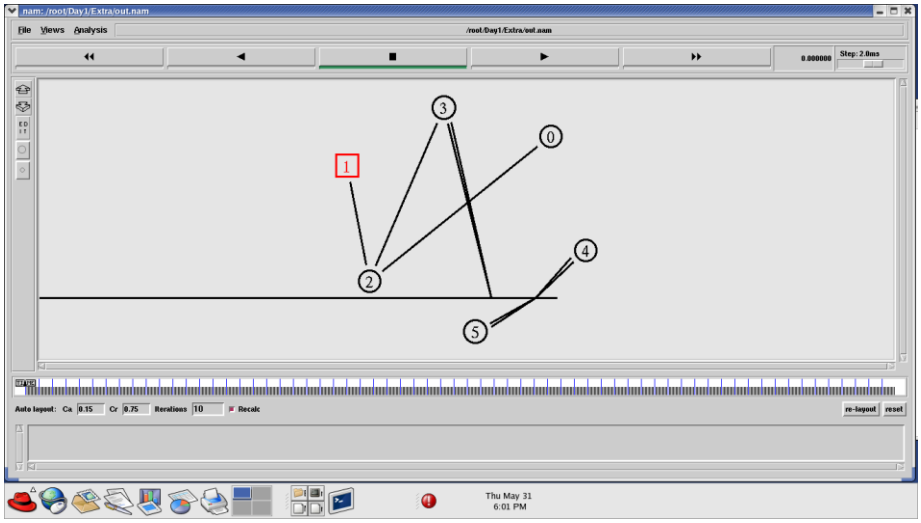
#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10

#Setup a TCP connection
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetSize_ 552

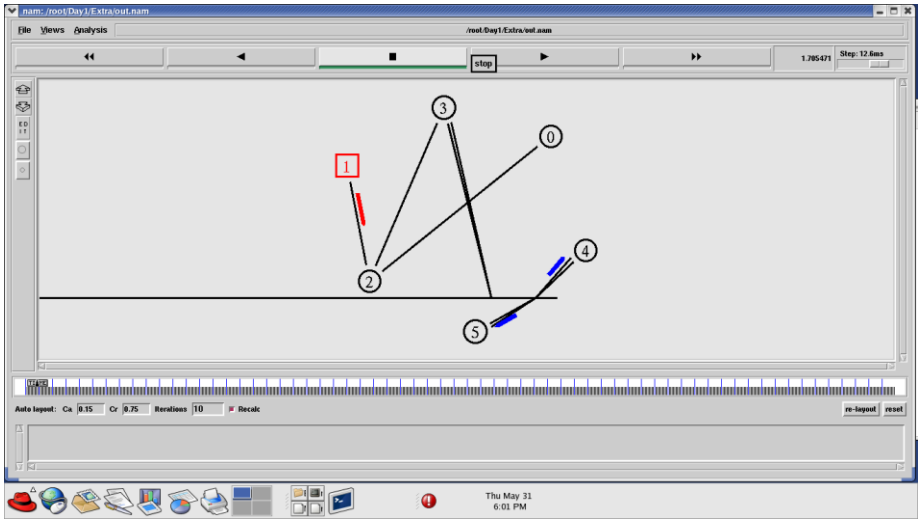
```
#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP


#Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null
$udp set fid_ 2

#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 0.01mb
$cbr set random_ false

$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 124.0 "$ftp stop"
$ns at 124.5 "$cbr stop"

# next procedure gets two arguments: the name of the
# tcp source node, will be called here "tcp",
# and the name of output file.

proc plotWindow {tcpSource file} {
global ns
set time 0.1
set now [$ns now]


set cwnd [$tcpSource set cwnd_]
set wnd [$tcpSource set window_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file" }
$ns at 0.1 "plotWindow $tcp $winfile"


$ns at 5 "$ns trace-annotate \"packet drop\""


# PPP

$ns at 125.0 "finish"
$ns run
```

OUTPUT

NETWORK FORMATION



DATA TRANSMISSION



4.2 UNICAST PROGRAM

```
set ns [new Simulator]

#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the Trace file
set file1 [open unicastDV.tr w]
$ns trace-all $file1

#Open the NAM trace file
set file2 [open unicastDV.nam w]
$ns namtrace-all $file2

#Define a 'finish' procedure
proc finish {} {
        global ns file1 file2
        $ns flush-trace
        close $file1
        close $file2
        exec nam unicastDV.nam &
        exit 0
}

# Next line should be commented out to have the static routing
$ns rtproto DV

#Create six nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
```

#Create links between the nodes

```
$ns duplex-link $n0 $n1 0.3Mb 10ms DropTail
$ns duplex-link $n1 $n2 0.3Mb 10ms DropTail
$ns duplex-link $n2 $n3 0.3Mb 10ms DropTail
$ns duplex-link $n1 $n4 0.3Mb 10ms DropTail
$ns duplex-link $n3 $n5 0.5Mb 10ms DropTail
$ns duplex-link $n4 $n5 0.5Mb 10ms DropTail
```

#Give node position (for NAM)

```
$ns duplex-link-op  $n0 $n1 orient right
$ns duplex-link-op  $n1 $n2 orient right
$ns duplex-link-op $n2 $n3 orient up
$ns duplex-link-op $n1 $n4 orient up-left
$ns duplex-link-op  $n3 $n5 orient left-up
$ns duplex-link-op  $n4 $n5 orient right-up
```

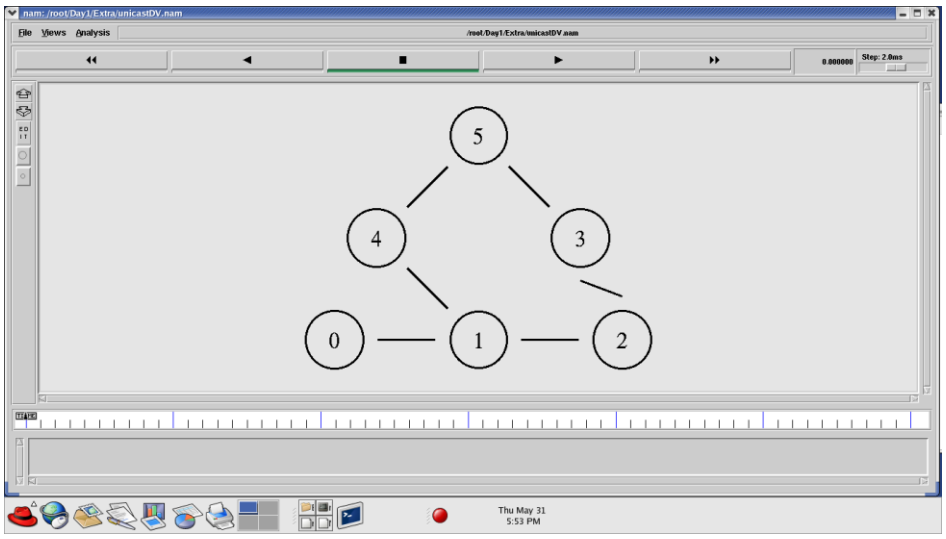
#Setup a TCP connection

```
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n5 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
```

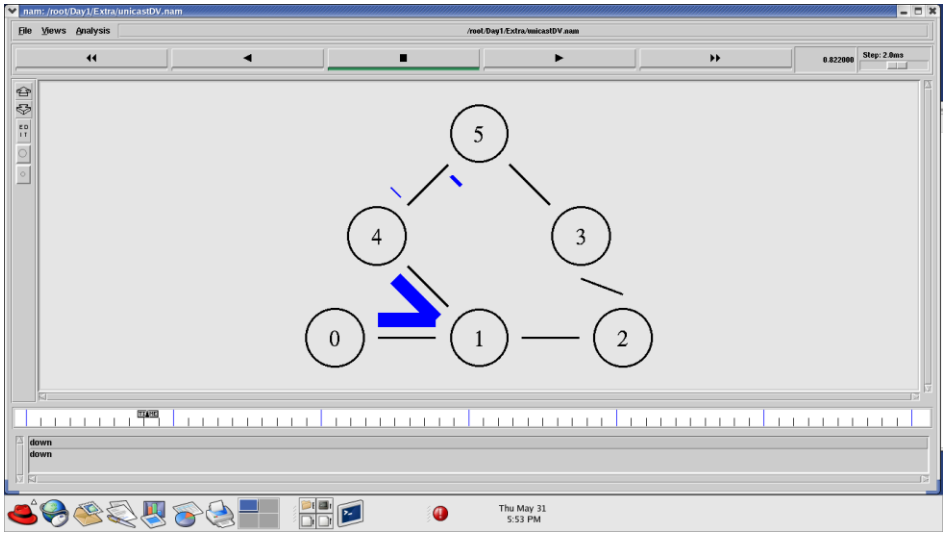
#Setup a FTP over TCP connection

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

$ns rtmodel-at 1.0 down $n1 $n4
$ns rtmodel-at 4.5 up $n1 $n4
$ns at 0.1 "$ftp start"
$ns at 6.0 "finish"
$ns run
```

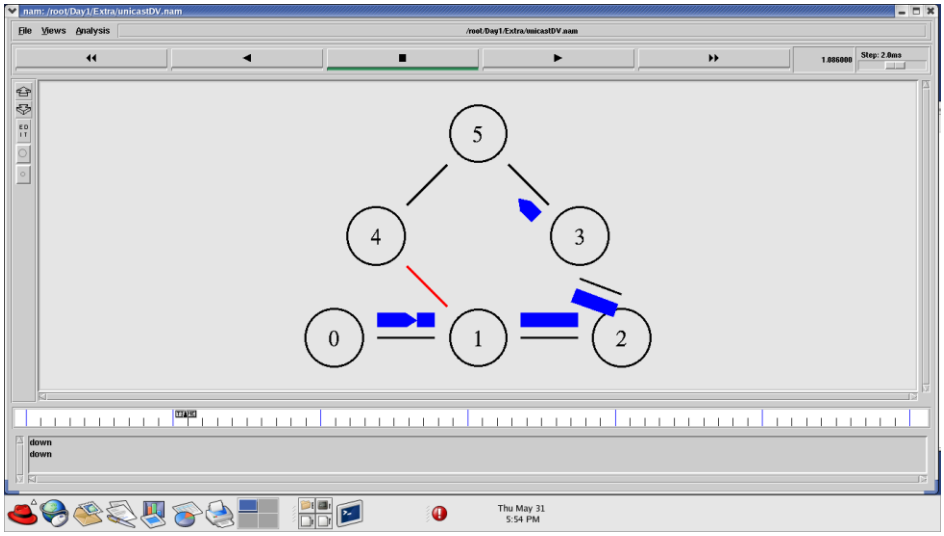
OUTPUT



DATA TRANSFER



PATH CHANGE DUE TO LINK FAILURE

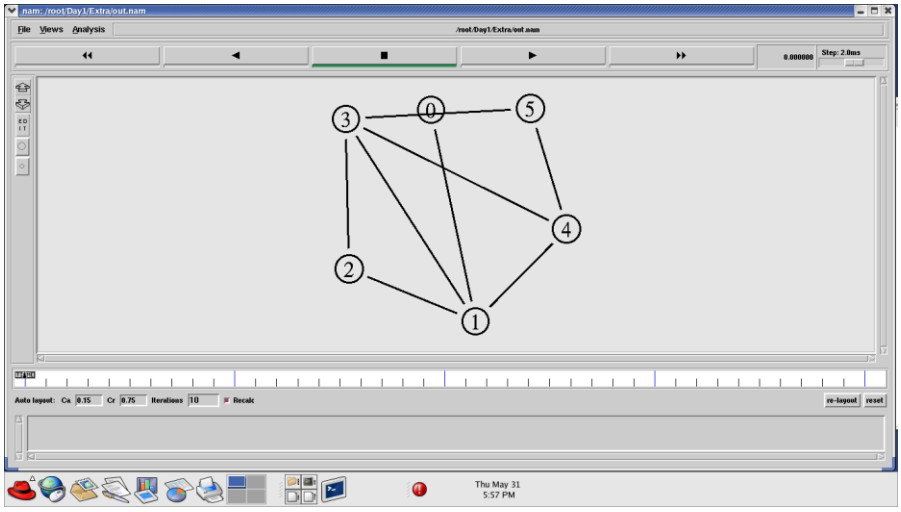


4.3 MULTICAST PROGRAM 1

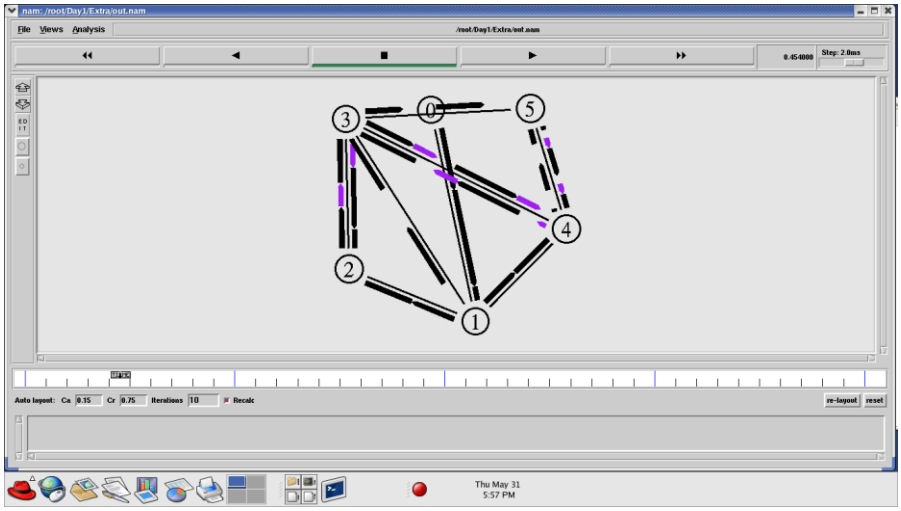
```
set ns [new Simulator]
$ns multicast

set f [open out.tr w]
$ns trace-all $f
$ns namtrace-all [open out.nam w]

$ns color 1 red
# the nam colors for the prune packets
$ns color 30 purple
# the nam colors for the graft packets
$ns color 31 green

# allocate a multicast address;
set group [Node allocaddr]

# nod is the number of nodes
set nod 6

# create multicast capable nodes;
for {set i 1} {$i <= $nod} {incr i} {
  set n($i) [$ns node]
}

#Create links between the nodes
$ns duplex-link $n(1) $n(2) 0.3Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) 0.3Mb 10ms DropTail
$ns duplex-link $n(2) $n(4) 0.5Mb 10ms DropTail
$ns duplex-link $n(2) $n(5) 0.3Mb 10ms DropTail
$ns duplex-link $n(3) $n(4) 0.3Mb 10ms DropTail
$ns duplex-link $n(4) $n(5) 0.5Mb 10ms DropTail
$ns duplex-link $n(4) $n(6) 0.5Mb 10ms DropTail
$ns duplex-link $n(5) $n(6) 0.5Mb 10ms DropTail
```

```
# configure multicast protocol;
set mproto DM

# all nodes will contain multicast protocol agents;
set mrthandle [$ns mrtproto $mproto]

set udp1 [new Agent/UDP]
set udp2 [new Agent/UDP]

$ns attach-agent $n(1) $udp1
$ns attach-agent $n(2) $udp2

set src1 [new Application/Traffic/CBR]
$src1 attach-agent $udp1
$udp1 set dst_addr_ $group
$udp1 set dst_port_ 0
$src1 set random_ false


set src2 [new Application/Traffic/CBR]
$src2 attach-agent $udp2
$udp2 set dst_addr_ $group
$udp2 set dst_port_ 1
$src2 set random_ false

# create receiver agents
set rcvr [new Agent/LossMonitor]

# joining and leaving the group;
$ns at 0.6 "$n(3) join-group $rcvr $group"
$ns at 1.3 "$n(4) join-group $rcvr $group"
$ns at 1.6 "$n(5) join-group $rcvr $group"
$ns at 1.9 "$n(4) leave-group $rcvr $group"
$ns at 2.3 "$n(6) join-group $rcvr $group"


 $ns at 3.5 "$n(3) leave-group $rcvr $group"

 $ns at 0.4 "$src1 start"
 $ns at 2.0 "$src2 start"

 $ns at 4.0 "finish"

 proc finish {} {
        global ns
        $ns flush-trace
        exec nam out.nam &
        exit 0
 }

 $ns run
```

OUTPUT



DATA TRANSMISSION

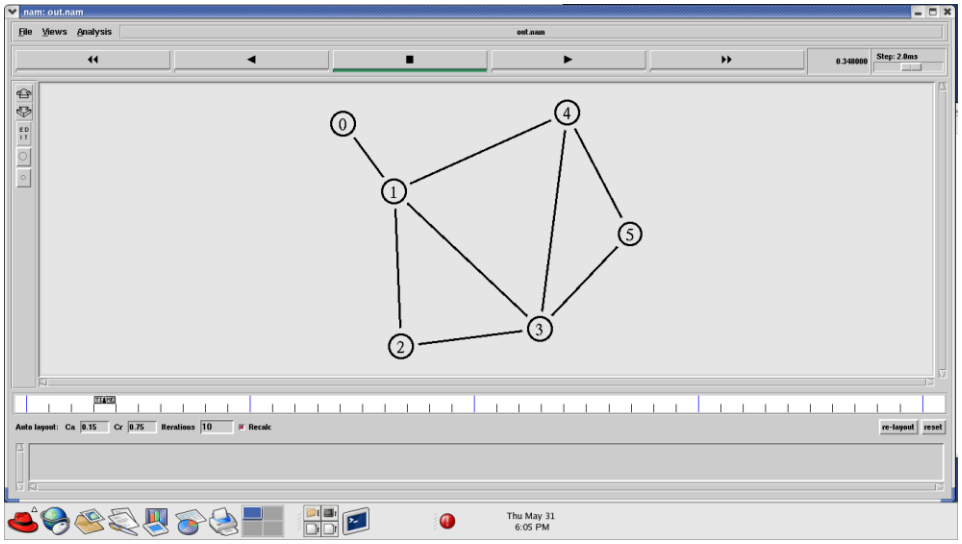


4.4 MULTICAST PROGRAM 2

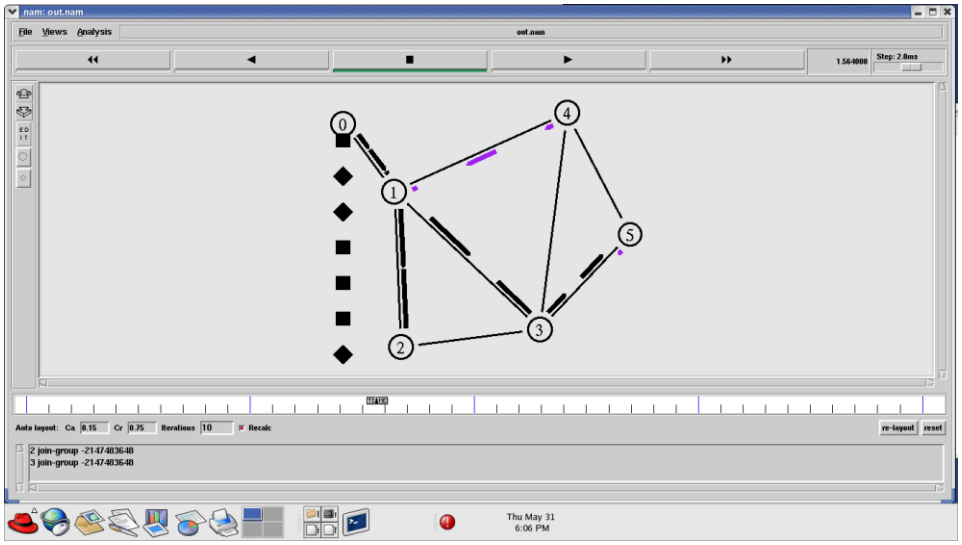
```
set ns [new Simulator]
$ns multicast

set f [open out.tr w]
$ns trace-all $f
$ns namtrace-all [open out.nam w]

$ns color 1 red
# the nam colors for the prune packets
$ns color 30 purple
# the nam colors for the graft packets
$ns color 31 green

# allocate a multicast address;
set group [Node allocaddr]

# nod is the number of nodes
set nod 6

# create multicast capable nodes;
for {set i 1} {$i <= $nod} {incr i} {
  set n($i) [$ns node]
}

#Create links between the nodes
$ns duplex-link $n(1) $n(2) 0.3Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) 0.3Mb 10ms DropTail
$ns duplex-link $n(2) $n(4) 0.5Mb 10ms DropTail
$ns duplex-link $n(2) $n(5) 0.3Mb 10ms DropTail
$ns duplex-link $n(3) $n(4) 0.3Mb 10ms DropTail
$ns duplex-link $n(4) $n(5) 0.5Mb 10ms DropTail
$ns duplex-link $n(4) $n(6) 0.5Mb 10ms DropTail
$ns duplex-link $n(5) $n(6) 0.5Mb 10ms DropTail
```

```
# configure multicast protocol;
DM set CacheMissMode dvmrp
set mproto DM

# all nodes will contain multicast protocol agents;
set mrthandle [$ns mrtproto $mproto]

set udp1 [new Agent/UDP]
set udp2 [new Agent/UDP]

$ns attach-agent $n(1) $udp1
$ns attach-agent $n(2) $udp2

set src1 [new Application/Traffic/CBR]
$src1 attach-agent $udp1
$udp1 set dst_addr_ $group
$udp1 set dst_port_ 0
$src1 set random_ false


set src2 [new Application/Traffic/CBR]
$src2 attach-agent $udp2
$udp2 set dst_addr_ $group
$udp2 set dst_port_ 1
$src2 set random_ false

# create receiver agents
set rcvr [new Agent/LossMonitor]

# joining and leaving the group;
$ns at 0.6 "$n(3) join-group $rcvr $group"
$ns at 1.3 "$n(4) join-group $rcvr $group"
$ns at 1.6 "$n(5) join-group $rcvr $group"
$ns at 1.9 "$n(4) leave-group $rcvr $group"


$ns at 2.3 "$n(6) join-group $rcvr $group"
$ns at 3.5 "$n(3) leave-group $rcvr $group"

$ns at 0.4 "$src1 start"
$ns at 2.0 "$src2 start"

$ns at 4.0 "finish"

proc finish {} {
        global ns
        $ns flush-trace
        exec nam out.nam &
        exit 0
}

$ns run
```

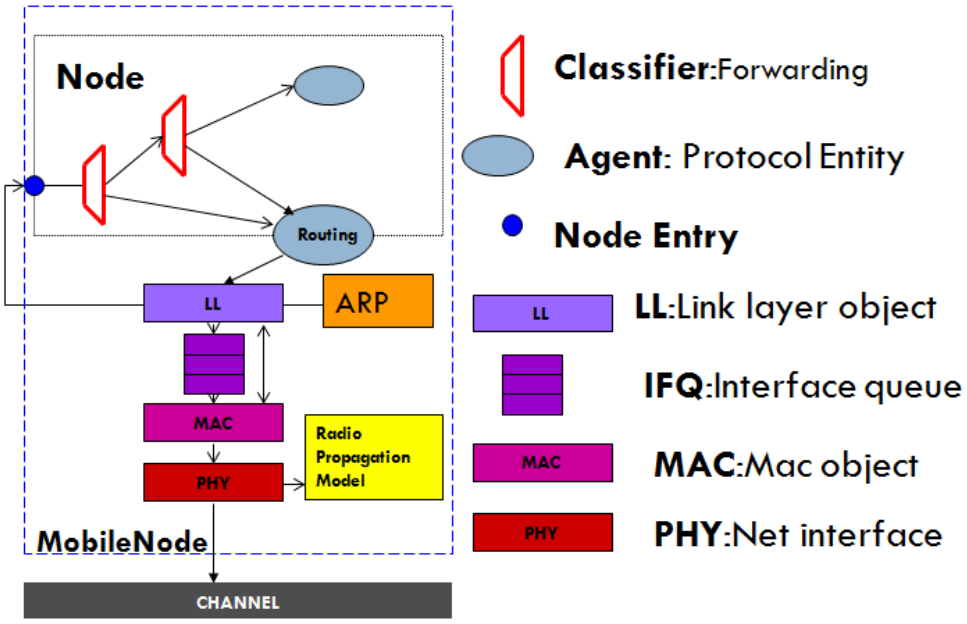
OUTPUT



DATA TRANSMISSION



4.5 Mobile/Wireless Node Structure



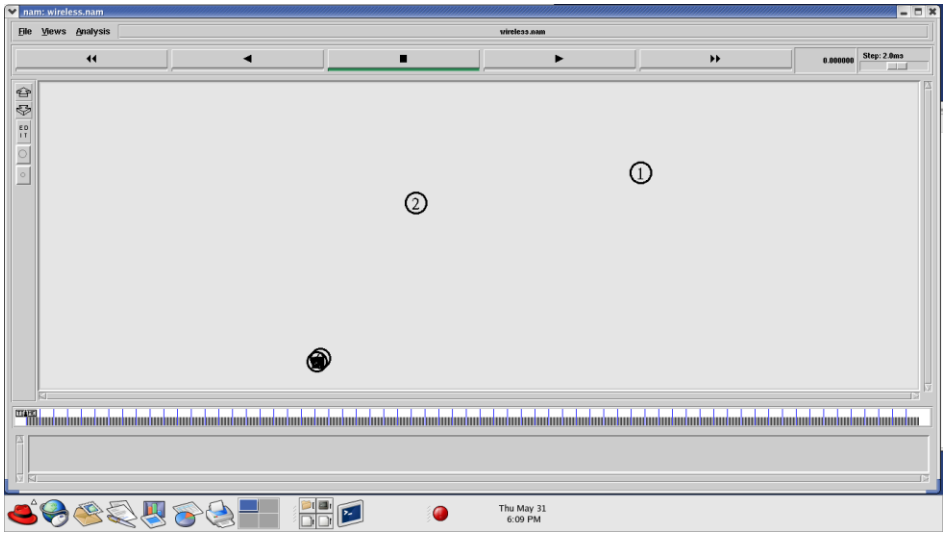
4.6 WIRELESS PROGRAM 1

```
# Define options
set val(chan)         Channel/WirelessChannel     ;# channel type
set val(prop)         Propagation/TwoRayGround    ;# radio-propagation
set val(netif)        Phy/WirelessPhy             ;# network interface
set val(mac)          Mac/802_11                  ;# MAC type
set val(ifq)          Queue/DropTail/PriQueue     ;# interface queue
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)           20                          ;# number of nodes
set val(rp)           AODV                        ;# routing protocol
set val(x)            500                         ;# X dimension
set val(y)            400                         ;# Y dimension
set val(stop)         110                         ;# simulation end

set ns              [new Simulator]
set tracefd     [open wireless.tr w]
set namtrace    [open wireless.nam w]

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

# 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) \
```

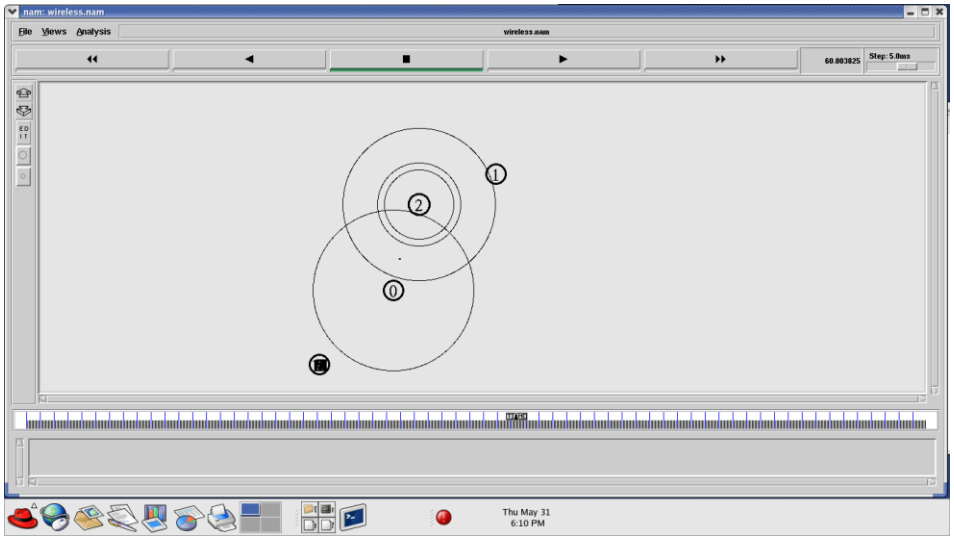
```
                -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -antType $val(ant) \
                -propType $val(prop) \
                -phyType $val(netif) \
                -channelType $val(chan)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

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

# Provide initial location of nodes
$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 udp [new Agent/UDP]
set sink [new Agent/Null]
$ns attach-agent $node_(0) $udp
$ns attach-agent $node_(1) $sink
$ns connect $udp $sink
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set interval_ 1
$cbr set maxpkts_ 100
$ns at 10.0 "$cbr 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 110.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
}

$ns run
```

OUTPUT



DATA TRANSFER



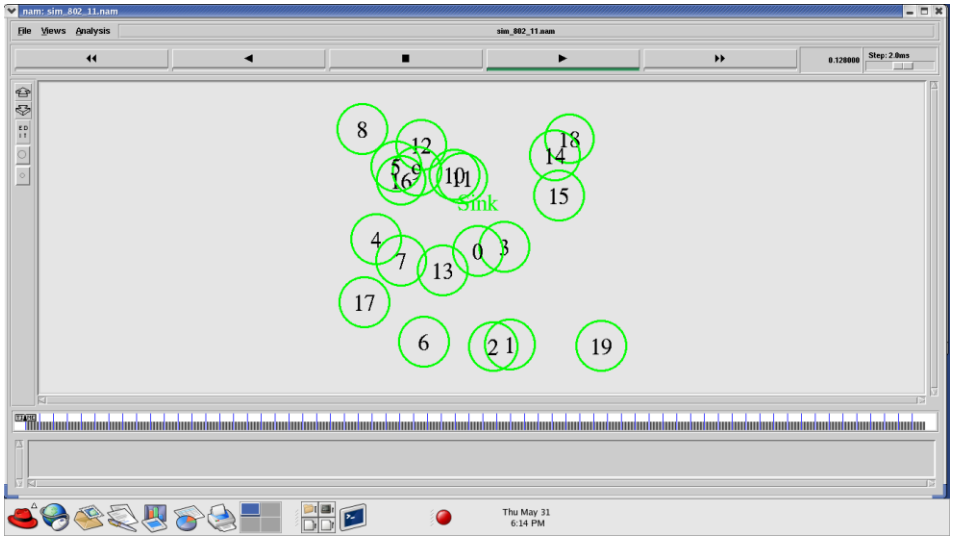
4.7 WSN PROGRAM – 802.11

```
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation
set val(netif)          Phy/WirelessPhy            ;# network interface
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)         100                        ;# max packet in ifq
set val(nn)             20                         ;# number of nodes
set val(rp)             AODV                       ;# protocol type
set val(x)              50                         ;# X dimension
set val(y)              50                         ;# Y dimension
set val(stop)           500                        ;# simulation period
set val(energymodel)    EnergyModel                ;# Energy Model
set val(initialenergy)  100                        ;# value

set ns              [new Simulator]
set tracefd         [open sim_802_11.tr w]
set namtrace        [open sim_802_11.nam w]
$ns use-newtrace
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
# set up topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)
# 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) \
```

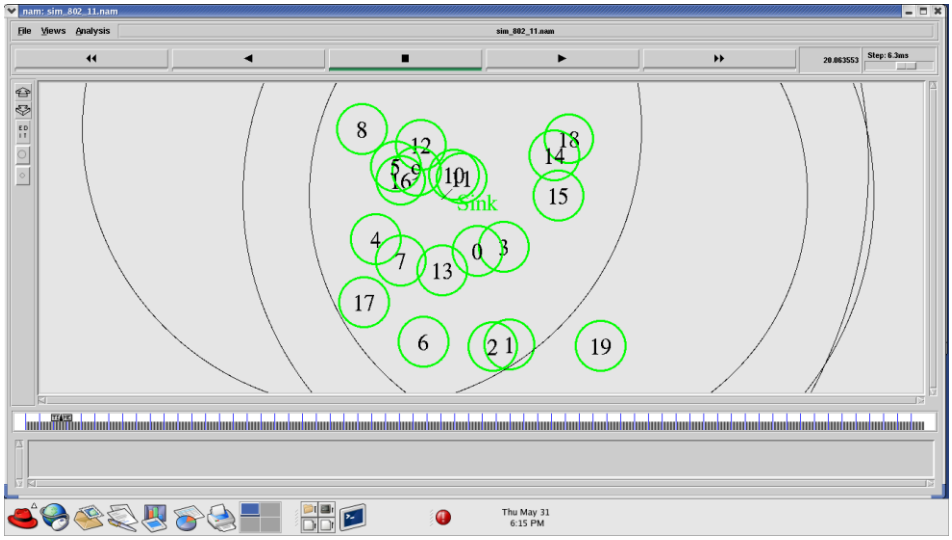
```
            -channel [new $val(chan)] \
            -topoInstance $topo \
            -agentTrace OFF \
            -routerTrace OFF \
            -macTrace  ON \
            -movementTrace OFF \
            -energyModel $val(energymodel) \
            -initialEnergy  $val(initialenergy) \
            -rxPower 35.28e-3 \
            -txPower 31.32e-3 \
             -idlePower  712e-6 \
             -sleepPower  144e-9
for {set i 0} {$i < $val(nn) } { incr i } {
       set mnode_($i) [$ns node]
}
for {set i 1} {$i < $val(nn) } { incr i } {
        $mnode_($i) set X_ [ expr {$val(x) * rand()} ]
        $mnode_($i) set Y_ [ expr {$val(y) * rand()} ]
        $mnode_($i) set Z_ 0
}
# Position of Sink
$mnode_(0) set X_ [ expr {$val(x)/2} ]
$mnode_(0) set Y_ [ expr {$val(y)/2} ]
$mnode_(0) set Z_ 0.0
$mnode_(0) label "Sink"
for {set i 0} {$i < $val(nn)} { incr i } {
        $ns initial_node_pos $mnode_($i) 10
}
#Setup a UDP connection
for {set i 1} {$i < $val(nn) } { incr i } {
set udp($i) [new Agent/UDP]
$ns attach-agent $mnode_($i) $udp($i)
}
set sink [new Agent/Null]
$ns attach-agent $mnode_(0) $sink
```

```
for {set i 1} {$i < $val(nn) } { incr i } {
$ns connect $udp($i) $sink
}
#Setup a CBR over UDP connection
for {set i 1} {$i < $val(nn) } { incr i } {
set cbr($i) [new Application/Traffic/CBR]
$cbr($i) attach-agent $udp($i)
$cbr($i) set type_ CBR
$cbr($i) set packet_size_ 100
$cbr($i) set maxpkts_ 100
#$cbr($i) set rate_ 0.1Mb
$cbr($i) set interval_ 1
$cbr($i) set random_ false
}
for {set i 1} {$i < $val(nn) } { incr i } {
$ns at [expr {$i + 5}] "$cbr($i) start"
}
for {set i 1} {$i < $val(nn) } { incr i } {
$ns at [expr $val(stop) - $i] "$cbr($i) stop"
}
# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
   $ns at $val(stop) "$mnode_($i) reset;"
}
# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at [expr $val(stop) + 0.01] "puts \"end simulation\"; $ns halt"
proc stop {} {
   global ns tracefd namtrace
   $ns flush-trace
   close $tracefd
   close $namtrace
}
$ns run
```

OUTPUT



DATA TRANSFER



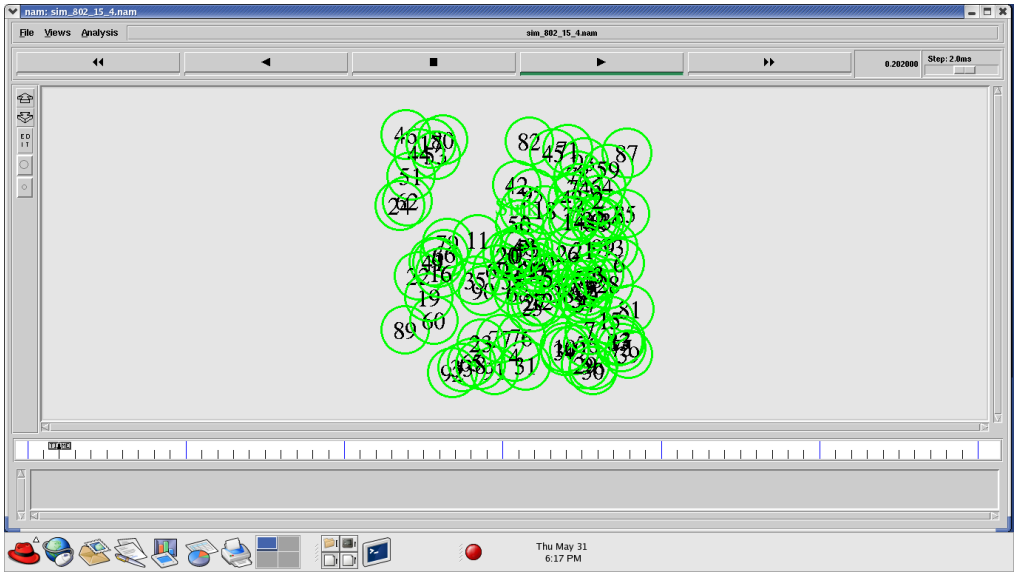
4.8 WSN PROGRAM – 802.15.4

```
set val(chan)       Channel/WirelessChannel    ;# channel type
set val(prop)       Propagation/TwoRayGround ;# radio-propagation
set val(netif)      Phy/WirelessPhy/802_15_4  ;# network interface
set val(mac)        Mac/802_15_4               ;# 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)     100                        ;# max packet in ifq
set val(nn)         100                        ;# number of nodes
set val(rp)         AODV                       ;# protocol type
set val(x)          50                         ;# X dimension
set val(y)          50                         ;# Y dimension
set val(stop)       500                        ;# simulation period
set val(energymodel)  EnergyModel              ;# Energy Model
set val(initialenergy)  100                    ;# value
set ns              [new Simulator]
set tracefd         [open sim_802_15_4.tr w]
set namtrace        [open sim_802_15_4.nam w]
$ns use-newtrace
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
# set up topography object
set topo      [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)
# 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) \
          -channel [new $val(chan)] \
```

```
            -topoInstance $topo \
            -agentTrace OFF \
            -routerTrace OFF \
            -macTrace  ON \
            -movementTrace OFF \
            -energyModel $val(energymodel) \
            -initialEnergy $val(initialenergy) \
            -rxPower 35e-3 \
            -txPower 31e-3 \
             -idlePower 31e-3 \
             -sleepPower 15e-9
for {set i 0} {$i < $val(nn) } { incr i } {
      set mnode_($i) [$ns node]
}
for {set i 1} {$i < $val(nn) } { incr i } {
       $mnode_($i) set X_ [ expr {$val(x) * rand()} ]
       $mnode_($i) set Y_ [ expr {$val(y) * rand()} ]
       $mnode_($i) set Z_ 0
}
# Position of Sink
$mnode_(0) set X_ [ expr {$val(x)/2} ]
$mnode_(0) set Y_ [ expr {$val(y)/2} ]
$mnode_(0) set Z_ 0.0
$mnode_(0) label "Sink"
for {set i 0} {$i < $val(nn)} { incr i } {
       $ns initial_node_pos $mnode_($i) 10
}
#Setup a UDP connection
for {set i 1} {$i < $val(nn) } { incr i } {
set udp($i) [new Agent/UDP]
$ns attach-agent $mnode_($i) $udp($i)
}
set sink [new Agent/Null]
$ns attach-agent $mnode_(0) $sink
```

```
for {set i 1} {$i < $val(nn) } { incr i } {
$ns connect $udp($i) $sink
}
#Setup a CBR over UDP connection
for {set i 1} {$i < $val(nn) } { incr i } {
set cbr($i) [new Application/Traffic/CBR]
$cbr($i) attach-agent $udp($i)
$cbr($i) set type_ CBR
$cbr($i) set packet_size_ 100
$cbr($i) set maxpkts_ 100
#$cbr($i) set rate_ 0.1Mb
$cbr($i) set interval_ 1
$cbr($i) set random_ false
}
for {set i 1} {$i < $val(nn) } { incr i } {
$ns at [expr {$i + 5}] "$cbr($i) start"
}
for {set i 1} {$i < $val(nn) } { incr i } {
$ns at [expr $val(stop) - $i] "$cbr($i) stop"
}
# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
   $ns at $val(stop) "$mnode_($i) reset;"
}
# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at [expr $val(stop) + 0.01] "puts \"end simulation\"; $ns halt"
proc stop {} {
   global ns tracefd namtrace
   $ns flush-trace
   close $tracefd
   close $namtrace
}
$ns run
```

OUTPUT



5 Protocol Works

5.1 Procedure to construct Malicious Node in TCL Script and C++

- Modification in AODV PROTOCOL
- LOCATION – ns-allinone-2.33/ns2.33/aodv/aodv.cc
- LOCATION – ns-allinone-2.33/ns2.33/aodv/aodv.h

aodv.h file changes

Declare a boolean variable malicious as shown below in the protected scope in the class AODV

bool malicious;

aodv.cc file changes

1. Initialize the **malicious** variable with a value "false". Declare it inside the constructor as shown below

AODV::AODV(nsaddr_t id):Agent(PT_AODV)...

{

.....

malicious = false;

}

2. Add the following statement to the aodv.cc file in the "if(argc==2)" statement.

```
if(strcmp(argv[1], "malicious") == 0) {
  desyn = true;
  return TCL_OK;
}
```

3. Implement the behavior of the **malicious** node by setting the following code in the rt_resolve(Packet *p) function. The **malicious** s node will simply drop the packet as indicated below.

if(malicious ==true)

{

drop(p,DROP_RTR_ROUTE_LOOP);

}

Once done, recompile ns2 as given below

Open Terminal -> Go to ~ns-2.33/ directory and type the command make to compile

$] cd /ns-allinone-2.33/ns-2.33/

$] make

Once the compilation is done, Check the **malicious** behavior using the Tcl Script by setting any four node as **malicious** node. The command to set the **malicious** node is

$ns at 2.0 "[$n0 set ragent_] malicious "

$ns at 2.0 "[$n8 set ragent_] malicious "

$ns at 2.0 "[$n23 set ragent_] malicious "

$ns at 2.0 "[$n19 set ragent_] malicious "

5.2 How to generate random mobility in ns2?

Procedure

Open the new terminal

cd ns-allinone-2.34

cd ns-2.34

cd indep-utils

pwd

ls

cd cmu-scen-gen

ls

cd setdest

ls

./setdest

./setdest –v 2 –n 10 –s 1 –m 10 –M 50 –t 30 –P 1 –p 1 –x 500 –y 500
./setdest –v 2 –n 10 –s 1 –m 10 –M 50 –t 30 –P 1 –p 1 –x 500 –y 500 >usersetdest.tcl
gedit usersetdest.tcl

5.3 How to generate random agent and application creation in ns2?

Procedure

cd ns-allinone-2.34
cd ns-2.34
cdindep-utils
cdcmu-scen-gen
ls
nscbrgen.tcl
nscbrgen.tcl –type cbr –nn 10 –seed 1 –mc 5 –rate 5.0
nscbrgen.tcl –type cbr –nn 10 –seed 1 –mc 5 –rate 5.0 > cbr-10.tcl
gedit cbr-10.tcl

6. PROGRAMS

6.1 PROGRAMS 1 – Wireless Network Construction using TCL script

Program Description

Basic wirless construction with number of nodes contained is three. The procedure to create nam file and trace file is given in this program. Topology is created by giving the position to the nodes and is specified by X, Y and Z coordinates. Here initial size of each and every nodes are built using initial_node_pos.The routing protocol which is used in this program is AODV (Adhoc On-demand Vector Routing Protocol). And simulation end time is 10ms.

File Name – program1.tcl

- Channel Type – Wireless Channel
- Propogation – Two Ray Ground Model
- X dimension – 500
- Y dimension – 400

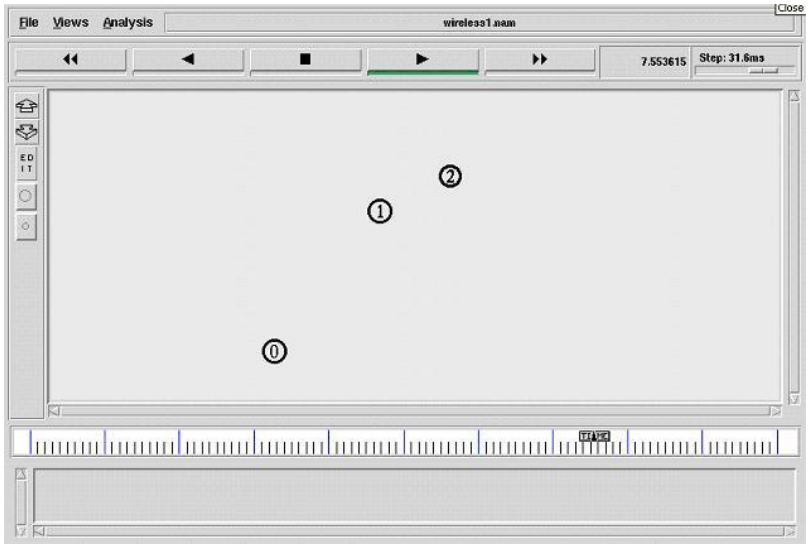
```
# Define options
set val(chan)     Channel/WirelessChannel  ;# channel type
set val(prop)     Propagation/TwoRayGround ;# radio-propagation model
set val(netif)    Phy/WirelessPhy          ;# network interface type
set val(mac)      Mac/802_11               ;# MAC type
set val(ifq)      Queue/DropTail/PriQueue  ;# 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)       3                        ;# number of mobilenodes
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)     10                       ;# time of simulation end
```

```
#-------Event scheduler object creation--------#

set ns [new Simulator]

# Creating trace file and nam file

set tracefd [open wireless1.tr w]
set namtrace [open wireless1.nam w]

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

# set up topography object
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]

# 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

## Creating node objects...
for {set i 0} {$i < $val(nn) } { incr i } {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < $val(nn)  } {incr i } {
        $node_($i) color black
        $ns at 0.0 "$node_($i) color black"
    }

# Provide initial location of mobile nodes
$node_(0) set X_ 50.0
```

```
$node_(0) set Y_ 50.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 200.0
$node_(1) set Y_ 250.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 300.0
$node_(2) set Y_ 300.0
$node_(2) set Z_ 0.0

# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
$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 10.01 "puts \"end simulation\"; $ns halt"
#stop procedure:
proc stop {} {
   global ns tracefd namtrace
   $ns flush-trace
   close $tracefd
   close $namtrace
exec nam wireless1.nam &
}

$ns run
```

Procedure to run the program in the terminal window - $ns program1.tcl

OUTPUT



6.2 PROGRAM 2 – Code for the construction of wireless nodes with fixed colors

Program Description:

Number of nodes in the network is eight which are created and configured as mobile wireless nodes. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is specified by using the X, Y and Z coordinates and the Z coordinates are always remains zero. Routing protocol is AODV and the stop time of the simulation is 10ms. Here all the nodes are created in cyan color.

- Channel Type – Wireless Channel
- Propogation – Two Ray Ground Model
- Queue Type – DropTail
- Antenna Type – Omni Directional Antenna
- Number of nodes – 8
- Routing protocol - AODV
- X dimension – 500
- Y dimension – 400
- Stop time – 10ms
- Color – cyan color

File Name – program2.tcl

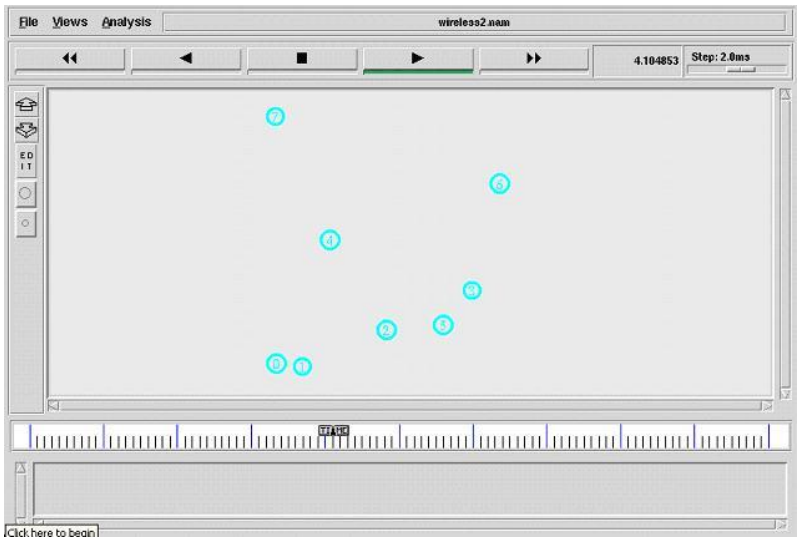
```
# Define options
set val(chan)     Channel/WirelessChannel    ;# channel type
set val(prop)     Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)    Phy/WirelessPhy            ;# network interface type
set val(mac)      Mac/802_11                 ;# MAC type
set val(ifq)      Queue/DropTail/PriQueue    ;# 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)       8                          ;# number of mobilenodes
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)     10                         ;# time of simulation end
```

```
#-------Event scheduler object creation------#

set ns          [new Simulator]

#Creating trace file and nam file.
set tracefd     [open wireless2.trw]
set namtrace    [open wireless2.nam w]

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

# set up topography object
set topo [new Topography]

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

set god_ [create-god $val(nn)]

# 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

# Creating node objects..
for {set i 0} {$i < $val(nn)} { incr i } {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < $val(nn) } {incr i} {
        $node_($i) color cyan
        $ns at 0.0 "$node_($i) color cyan"
    }
```

```
# Provide initial location of mobilenodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 30.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 50.0
$node_(1) set Y_ 25.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 200.0
$node_(2) set Y_ 90.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 350.0
$node_(3) set Y_ 160.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 100.0
$node_(4) set Y_ 250.0
$node_(4) set Z_ 0.0

$node_(5) set X_ 300.0
$node_(5) set Y_ 100.0
$node_(5) set Z_ 0.0

$node_(6) set X_ 400.0
$node_(6) set Y_ 350.0
$node_(6) set Z_ 0.0

$node_(7) set X_ 3s50.0
$node_(7) set Y_ 470.0
$node_(7) set Z_ 0.0

# 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 10.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec nam wireless2.nam &
}

$ns run
```

Procedure to run the program in the terminal window - $ns program2.tcl

OUTPUT



6.3 PROGRAM 3 – Dynamic node creation program using AODV protocol TCL script

Program Discription

Number of nodes in the network is not static in this program. Number of nodes construction is given during the run time of the program. The user should give the number of nodes in the terminal window during the execution of the program. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is specified by using the X, Y and Z coordinates and the Z coordinates are always remains zero. Routing protocol is AODV and the stop time of the simulation is 10ms. Here all the nodes are created in yellow color.

File Name – program3.tcl

```
if {$argc != 1} {
      error "\nCommand: ns wireless1.tcl <no.of.mobile-nodes>\n\n "
}
```

```
# 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)           [lindex $argv 0]           ;# number of mobilenodes
set val(rp)           AODV                       ;# routing protocol
set val(x)            600                        ;# X dimension of topograp
set val(y)            600                        ;# Y dimension of topograp
set val(stop)         10                         ;# time of simulation end
```

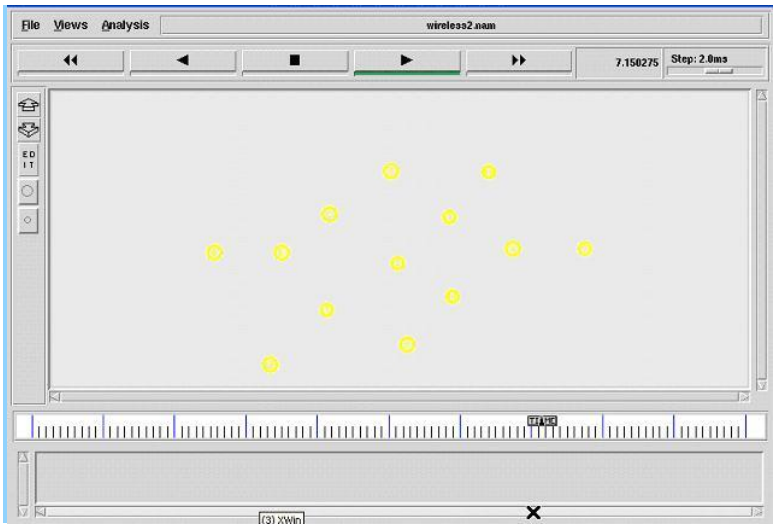
```
#-------Event scheduler object creation----#

set ns          [new Simulator]

#creating the trace file and nam file

set tracefd     [open wireless1.trw]
set namtrace    [open wireless1.nam w]

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

# set up topography object
set topo [new Topography]

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

set god_ [create-god $val(nn)]

# 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

## Creating node objects..
    for {set i 0} {$i < $val(nn)} {incr i} {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < $val(nn)} {incr i} {
        $node_($i) color yellow
        $ns at 0.0 "$node_($i) color yellow"
    }
```

```
# Provide initial location of mobilenodes
$node_(0) set X_ 27.0
    $node_(0) set Y_ 260.0
    $node_(0) set Z_ 0.0

    $node_(1) set X_ 137.0
    $node_(1) set Y_ 348.0
    $node_(1) set Z_ 0.0

    $node_(2) set X_ 294.0
    $node_(2) set Y_ 235.0
    $node_(2) set Z_ 0.0

    $node_(3) set X_ 414.0
    $node_(3) set Y_ 342.0
    $node_(3) set Z_ 0.0

    $node_(4) set X_ 562.0
    $node_(4) set Y_ 267.0
    $node_(4) set Z_ 0.0

    $node_(5) set X_ 279.0
    $node_(5) set Y_ 447.0
    $node_(5) set Z_ 0.0

    $node_(6) set X_ -128.0
    $node_(6) set Y_ 260.0
    $node_(6) set Z_ 0.0

$node_(7) set X_ 727.0
$node_(7) set Y_ 269.0
$node_(7) set Z_ 0.0

$node_(8) set X_ 130.0
$node_(8) set Y_ 126.0
$node_(8) set Z_ 0.0

$node_(9) set X_ 318.0
$node_(9) set Y_ 45.0
$node_(9) set Z_ 0.0
```

```
$node_(10) set X_ 505.0
$node_(10) set Y_ 446.0
$node_(10) set Z_ 0.0

$node_(11) set X_ 421.0
$node_(11) set Y_ 158.0
$node_(11) set Z_ 0.0


# 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 10.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec nam wireless1.nam &
}

$ns run
```

Procedure to run the program in the terminal window - $ns program3.tcl

OUTPUT



6.4 PROGRAM 4 – Dynamic node creation program and its initial location using AODV protocol TCL script

Program Discription

Number of nodes in the network is not static in this program. Number of nodes construction is given during the run time of the program. The user should give the number of nodes in the terminal window during the execution of the program. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is specified by using the X, Y and Z coordinates and the Z coordinates are always remains zero. Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is AODV and the stop time of the simulation is 10ms. Here all the nodes are created in yellow color.

File Name – program4.tcl

- X dimension – 600
- Y dimension – 600
- Stop time – 10ms
- Color – Yellow color
- Initial Node Position - 30

```
if {$argc != 1} {
      error "\nCommand: ns wireless3.tcl <no.of.mobile-nodes>\n\n "
}
```

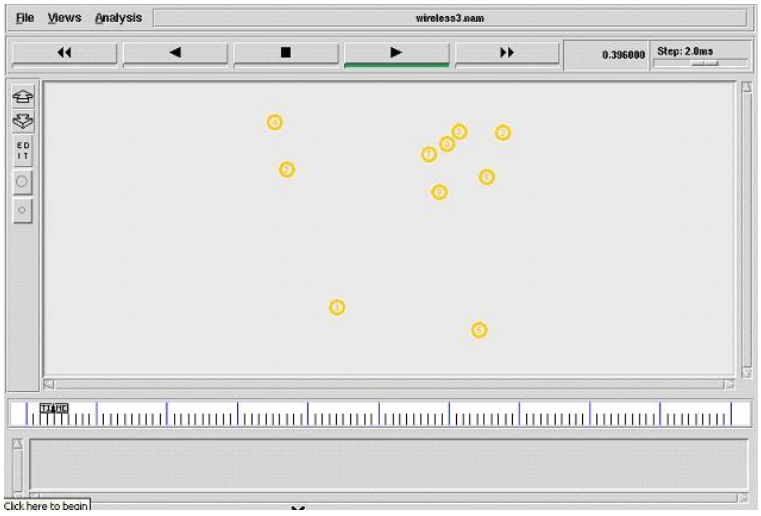
```
# Define options
set val(chan)     Channel/WirelessChannel    ;# channel type
set val(prop)     Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)    Phy/WirelessPhy            ;# network interface type
set val(mac)      Mac/802_11                 ;# MAC type
set val(ifq)      Queue/DropTail/PriQueue    ;# 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)       [lindex $argv 0]           ;# number of mobilenodes
set val(rp)       AODV                       ;# routing protocol
set val(x)        600                        ;# X dimension of topography
set val(y)        600                        ;# Y dimension of topography
set val(stop)     10                         ;# time of simulation end
```

```
#-------Event scheduler object creation------#

set ns [new Simulator]

#creating the trace file and nam file

set tracefd [open wireless3.trw]
set namtrace [open wireless3.namw]

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

# set up topography object
set topo       [new Topography]

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

set god_ [create-god $val(nn)]

# 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

## Creating node objects..
    for {set i 0} {$i < $val(nn)} { incr i } {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < $val(nn)} {incr i} {
        $node_($i) color gold
        $ns at 0.0 "$node_($i) color gold"
    }
```

```
## Provide initial location of mobilenodes..
        for {set i 0} {$i < $val(nn) } { incr i } {
            set xx [expr rand()*600]
            set yy [expr rand()*600]
            $node_($i) set X_ $xx
            $node_($i) set Y_ $yy

        }

# 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 10.01 "puts \"end simulation\" ; $ns halt"

#stop procedure...
proc stop {} {
   global ns tracefd namtrace
   $ns flush-trace
   close $tracefd
   close $namtrace
exec nam wireless3.nam &
}

$ns run
```

Procedure to run the program in the terminal window - $ns program4.tcl

OUTPUT



6.5 PROGRAM 5 – Dynamic color creation program and its initial location of nodes using AODV routing protocol TCL script

Program Discription

Number of nodes in the network is static in this program. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is not static. X and Y coordinates are randomly selected and the Z coordinates are always remains zero. Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is AODV and the stop time of the simulation is 10ms. Here all the nodes colors will get modified dynamically according to the time period

File Name – program5.tcl

- Number of nodes - 4
- X dimension – 750
- Y dimension – 550
- Stop time – 3.0ms
- Color – Yellow color
- Initial Node Position - 30

```
##  Setting The wireless Channels..
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)  5                          ;# max packet in ifq
set val(nn)      4                          ;# number of mobilenodes
set val(rp)      AODV                       ;# routing protocol
set val(x)       750                        ;# X dimension of topography
set val(y)       550                        ;# Y dimension of topography
set val(stop)    3.0                        ;# time of simulation end
```

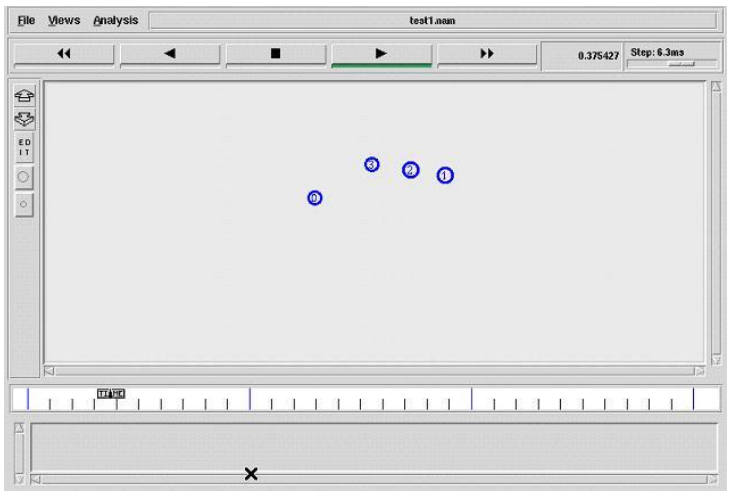
```
#-------Event scheduler object creation--------#

    set ns [new Simulator]

## Create a trace file and nam file..
    set tracefd [open wireless1.tr w]
    set namtrace [open wireless1.nam w]

## Trace the nam and trace details from the main simulation..
    $ns trace-all $tracefd
    $ns namtrace-all-wireless  $namtrace $val(x) $val(y)

## set up topography object..
    set topo [new Topography]

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

    set god_ [create-god $val(nn)]

## Color Descriptions..
    $ns color 1 dodgerblue
    $ns color 2 blue
    $ns color 3 cyan
    $ns color 4 green
    $ns color 5 yellow
    $ns color 6 black
    $ns color 7 magenta
    $ns color 8 gold
    $ns color 9 red

## Array for dynamic color settings...
    set colorname(0) blue
    set colorname(1) cyan
    set colorname(2) green
    set colorname(3) red
    set colorname(4) gold
    set colorname(5) magenta

## Setting The Distance Variables..
# 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(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)

## Setting node config event with set of inputs..
      $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


## Creating node objects...
    for {set i 0} {$i < $val(nn) } { incr i } {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < $val(nn) } {incr i } {
        $node_($i) color blue
        $ns at 0.0 "$node_($i) color blue"
    }

## Provide initial location of mobilenodes...
        for {set i 0} {$i < $val(nn) } { incr i } {
            set xx [expr rand()*600]
            set yy [expr rand()*500]
            $node_($i) set X_ $xx
            $node_($i) set Y_ $yy
            $node_($i) set Z_ 0.0
        }
```

```
## 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
    }

## Dynamic color procedure..
$ns at 0.0 "dynamic-color"
proc dynamic-color {} {
    global ns val node_ colorname
    set time 0.3
    set now [$ns now]
    set Rand [expr round(rand()*5)]
    for {set i 0} {$i < $val(nn) } {incr i } {
        $node_($i) color $colorname($Rand)
        $ns at $now "$node_($i) color $colorname($Rand)"
    }
    $ns at [expr $now+$time] "dynamic-color"
}
## stop procedure..
$ns at $val(stop) "stop"
proc stop {} {
   global ns tracefd namtrace
   $ns flush-trace
   close $tracefd
   close $namtrace
   puts "running nam..."
   exec nam wireless1.nam &
   exit 0
}

$ns runs
```

Procedure to run the program in the terminal window - $ns program5.tcl

OUTPUT



6.6 PROGRAM 6 – Node mobility construction program using DSR routing protocol TCL script

Program Discription

Number of nodes in the network is static. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Movement for each and every node is built with static speed and spectifed receiver address which is randomly generated and also the mobility will get change accoding to the time period. Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is DSR and the stop time of the simulation is 10ms.

File Name – program6.tcl

- X dimension – 750
- Y dimension – 550
- Stop time – 3.0ms
- Color – Yellow color
- Initial Node Position - 30

```
if {$argc != 1} {
    error "\nCommand: ns program6.tcl <no.of.mobile-nodes>\n\n "
}
```

```
## Setting The wireless Channels..
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)  5                          ;# max packet in ifq
set val(nn)      [lindex $argv 0]           ;# number of mobilenodes
set val(rp)      DSR                        ;# routing protocol
set val(x)       750                        ;# X dimension of topography
set val(y)       550                        ;# Y dimension of topography
set val(stop)    10.0                       ;# time of simulation end
```

```
#-------Event scheduler object creation--------#

    set ns [new Simulator]

## Create a trace file and nam file..
    set tracefd [open wireless2.tr w]
    set namtrace [open wireless2.nam w]

## Trace the nam and trace details from the main simulation..
    $ns trace-all $tracefd
    $ns namtrace-all-wireless $namtrace $val(x) $val(y)

## set up topography object..
    set topo [new Topography]

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

    set god_ [create-god $val(nn)]

## Color Descriptions..
    $ns color 1 dodgerblue
    $ns color 2 blue
    $ns color 3 cyan
    $ns color 4 green
    $ns color 5 yellow
    $ns color 6 black
    $ns color 7 magenta
    $ns color 8 gold
    $ns color 9 red
## Setting The Distance Variables..
# 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(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)
```

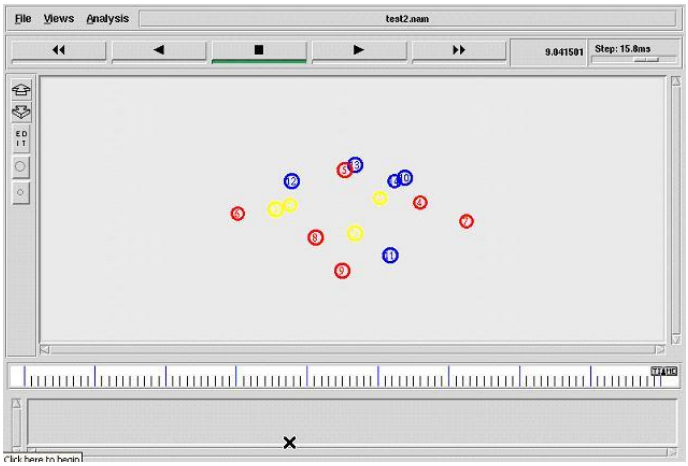
```
## Setting node config event with set of inputs..

      $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


## Creating node objects..
    for {set i 0} {$i < $val(nn) } { incr i } {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < 4 } {incr i } {
        $node_($i) color yellow
        $ns at 0.0 "$node_($i) color yellow"
    }
    for {set i 4} {$i < 10 } {incr i } {
        $node_($i) color red
        $ns at 3.0 "$node_($i) color red"
    }
    for {set i 10} {$i < 15 } {incr i } {
        $node_($i) color blue
        $ns at 5.0 "$node_($i) color blue"
    }

## Provide initial location of mobilenodes..
    $node_(0) set X_ 27.0
    $node_(0) set Y_ 260.0
    $node_(0) set Z_ 0.0

    $node_(1) set X_ 137.0
    $node_(1) set Y_ 348.0
    $node_(1) set Z_ 0.0
```

```
$node_(2) set X_ 294.0
$node_(2) set Y_ 235.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 414.0
$node_(3) set Y_ 342.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 562.0
$node_(4) set Y_ 267.0
$node_(4) set Z_ 0.0

$node_(5) set X_ 279.0
$node_(5) set Y_ 447.0
$node_(5) set Z_ 0.0
$node_(6) set X_ -128.0
$node_(6) set Y_ 260.0
$node_(6) set Z_ 0.0

$node_(7) set X_ 727.0
$node_(7) set Y_ 269.0
$node_(7) set Z_ 0.0

$node_(8) set X_ 130.0
$node_(8) set Y_ 126.0
$node_(8) set Z_ 0.0

$node_(9) set X_ 318.0
$node_(9) set Y_ 45.0
$node_(9) set Z_ 0.0

$node_(10) set X_ 505.0
$node_(10) set Y_ 446.0
$node_(10) set Z_ 0.0

$node_(11) set X_ 421.0
$node_(11) set Y_ 158.0
$node_(11) set Z_ 0.0

$node_(12) set X_ 72.0
$node_(12) set Y_ 397.0
$node_(12) set Z_ 0.0

if {$val(nn) >12} {
    for {set i 13} {$i < $val(nn)} { incr i } {
        set xx [expr rand()*600]
```

```
            set yy [expr rand()*500]
            $node_($i) set X_ $xx
            $node_($i) set Y_ $yy
            $node_($i) set Z_ 0.0
        }
    }
## 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
    }
## Stop procedure..
$ns at 0.0 "destination"
proc destination {} {
    global ns val node_
    set time 1.0
    set now [$ns now]
    for {set i 0} {$i<$val(nn)} {incr i} {
        set xx [expr rand()*600]
        set yy [expr rand()*500]
        $ns at $now "$node_($i) setdest $xx $yy 20.0"
    }
    $ns at [expr $now+$time] "destination"
}

$ns at $val(stop) "stop"

#stop procedure:
proc stop {} {
   global ns tracefd namtrace
   $ns flush-trace
   close $tracefd
   close $namtrace
   puts "running nam..."
   exec nam wireless2.nam &
   exit 0
}

$ns run
```

Procedure to run the program in the terminal window - $ns program6.tcl

OUTPUT



6.7 PROGRAM 7 – Creation of TCP (Transmission Control Protocol) communication between the nodes using AODV routing protocol TCL script

Program Discription

Number of nodes in the network is static and is declared as three in the network. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Movement for each and every node is built with static speed and spectifed receiver address which is randomly generated and also the mobility will get change accoding to the time period. Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is AODV and the stop time of the simulation is 150ms. Three nodes are created which are node0, node 1 and node 2. Send TCP agent is created and attached to node0, destination TCPsink agent is created and attached to node1. Then the TCP agent and the TCPsink agent are connected. In the next level, FTP application is created and attached to the sender TCP agent. Now the communication is initiated.

File Name – program7.tcl

- X dimension – 500
- Y dimension – 400
- Stop time – 150 ms

```
# 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)      3                          ;# number of mobilenodes
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
```

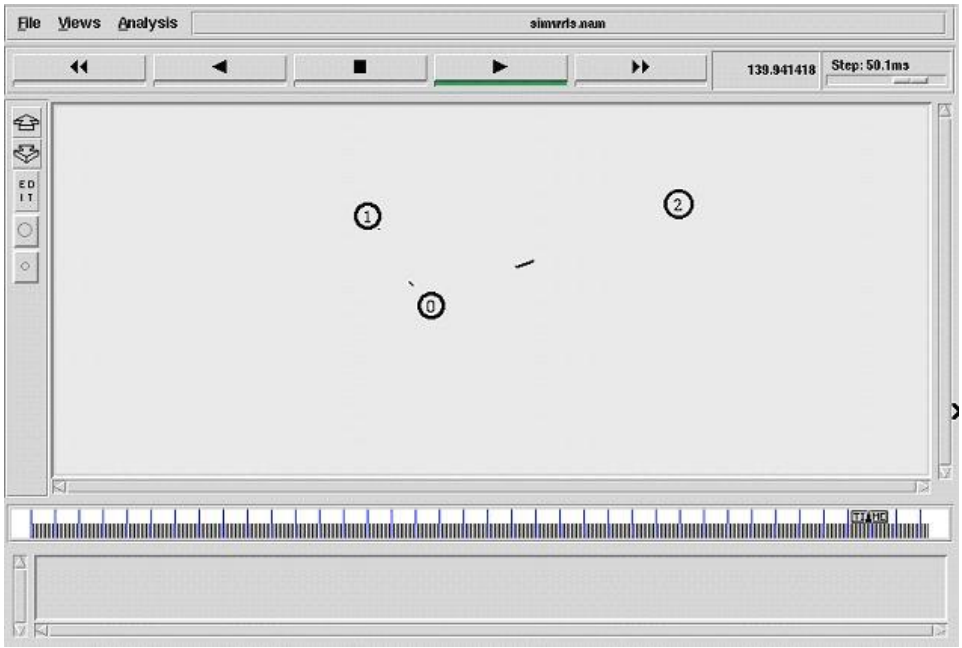
```
#-------Event scheduler object creation------#

set ns              [new Simulator]
#creating trace file and nam file
set tracefd       [open wireless1.trw]
set windowVsTime2 [open win.trw]
set namtrace      [open wireless1.nam w]

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

# set up topography object
set topo       [new Topography]

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

create-god $val(nn)

# 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_ 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"
$ns at 19.0 "$node_(2) 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"

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
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 10.0 "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
$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
exec nam simwrls.nam &
}

$ns run
```

Procedure to run the program in the terminal window - $ns program7.tcl

OUTPUT



6.8 PROGRAM 8 – Creation of TCP (Transmission Control Protocol) communication between the nodes using DSR routing protocol TCL script

Program Discription

Number of nodes in the network is static and is declared as three in the network. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Movement for each and every node is built with static speed and spectifed receiver address which is randomly generated and also the mobility will get change accoding to the time period. Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is DSR and the stop time of the simulation is 150ms. Three nodes are created which are node0, node 1 and node 2. Send TCP agent is created and attached to node0, destination TCPsink agent is created and attached to node1. Then the TCP agent and the TCPsink agent are connected. In the next level, FTP application is created and attached to the sender TCP agent. Now the communication is initiated.

File Name – program8.tcl

- X dimension – 500
- Y dimension – 400
- Stop time – 150 ms

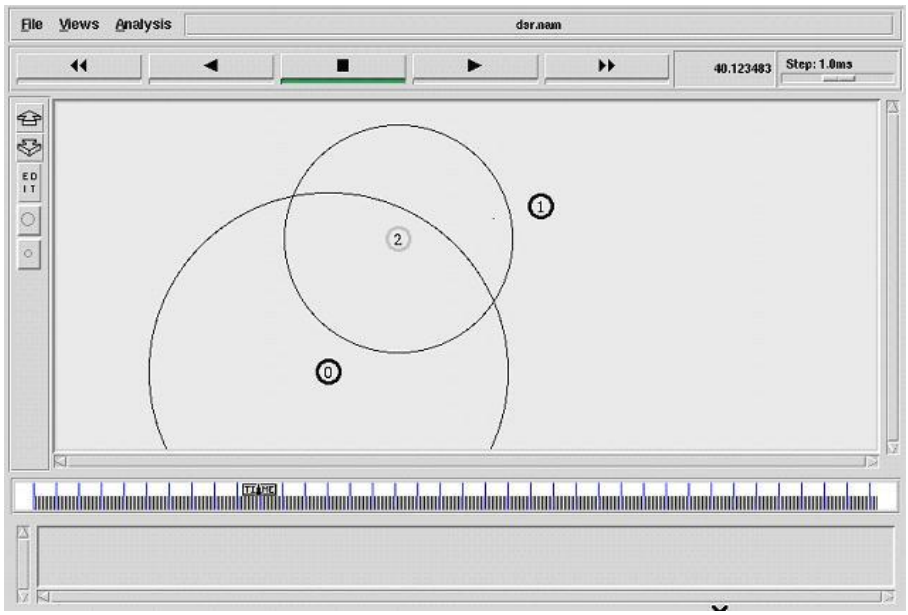
```
# Define options
set val(chan)     Channel/WirelessChannel    ;# channel type
set val(prop)     Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)    Phy/WirelessPhy            ;# network interface type
set val(mac)      Mac/802_11                 ;# MAC type
set val(ifq)      Queue/DropTail/PriQueue    ;# 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)       3                          ;# number of mobilenodes
set val(rp)       DSR                        ;# 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
```

```
#-------Event scheduler object creation------#

set ns              [new Simulator]
#Creating trace file and nam file
set tracefd       [open dsr.trw]
set windowVsTime2 [open win.trw]
set namtrace      [open dsr.nam w]

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

# set up topography object
set topo       [new Topography]

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

create-god $val(nn)

# 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_ 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"

# 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
$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
exec nam dsr.nam &
exit 0
}

$ns run
```

Procedure to run the program in the terminal window - $ns program8.tcl

OUTPUT



6.9 PROGRAM 9 – Creation of UDP (User Datagram Protocol) communication between nodes with CBR traffic using AODV routing protocol TCL script

Program Discription

Number of nodes in the network is static and is declared as 22 in the network. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Movement for each and every node is built with static speed and spectifed receiver address which is randomly generated and also the mobility will get change accoding to the time period. Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is AODV and the stop time of the simulation is 150ms. Send UDP agent is created and attached to sender node, destination UDPNull agent is created and attached to destination node. Then the UDP agent and the UDPNull agent are connected. In the next level, CBR application is created and attached to the sender UDP agent. Now the communication is initiated.

File Name – program9.tcl

- Number or nodes - 22
- X dimension – 1800
- Y dimension – 840
- Stop time – 150 ms

```
#Define setting option
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)        22                          ;# number of mobilenodes
set val(rp)        AODV                        ;# routing protocol
set val(x)         1800
set val(y)         840
```

```
### 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  $val(x) $val(y)

    set topo [new Topography]
    $topo load_flatgrid $val(x) $val(y)
    create-god $val(nn)
    set chan_1_ [new $val(chan)]

####  Setting The Distance Variables


    # 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)

#  Defining Node Configuration

            $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) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace ON \
                -movementTrace ON \
                -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
    }


### 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_ 741.0
$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 437.0 799.0 20.0"
$ns_ at 0.20 "$n20 setdest 159.0 722.0 20.0"
$ns_ at 1.25 "$n21 setdest 700.0 350.0 20.0"
$ns_ at 0.75 "$n22 setdest 839.0 444.0 20.0"
```

Setting The Node Size

```
$ns_ initial_node_pos $Server1 75
$ns_ initial_node_pos $Server2 75
$ns_ initial_node_pos $n2 40
$ns_ initial_node_pos $n3 40
$ns_ initial_node_pos $n4 40
$ns_ initial_node_pos $n5 40
$ns_ initial_node_pos $n6 40
$ns_ initial_node_pos $n7 40
$ns_ initial_node_pos $n8 40
$ns_ initial_node_pos $n9 40
$ns_ initial_node_pos $n10 40
$ns_ initial_node_pos $n11 40
$ns_ initial_node_pos $n12 40
$ns_ initial_node_pos $n13 40
$ns_ initial_node_pos $n14 40
$ns_ initial_node_pos $n15 40
$ns_ initial_node_pos $n16 40
$ns_ initial_node_pos $n17 40
$ns_ initial_node_pos $n18 40
$ns_ initial_node_pos $n19 40
```

```
    $ns_ initial_node_pos $n20 40
    $ns_ initial_node_pos $n21 40
    $ns_ initial_node_pos $n22 40


    #### Setting The Labels For Nodes

    $ns_ at 0.0 "$Server1 label Server1"
    $ns_ at 0.0 "$Server2 label Server2"

    #Setting Color For Server

    $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 15.0ms"

  #  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]
    $udp0 set fid_ 1
    set cbr0 [$udp0 attach-app Traffic/CBR]
```

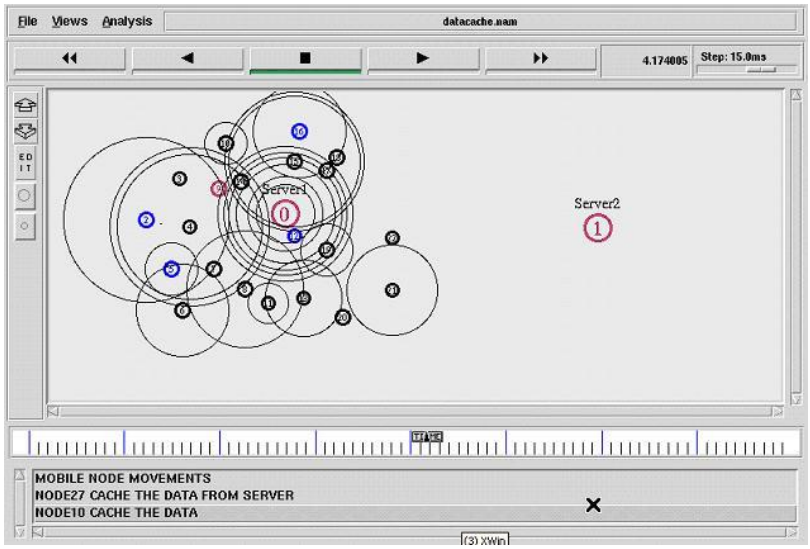
```
$cbr0 set packetSize_ 1000
$cbr0 set interval_ .07
$ns_ at 0.0 "$cbr0 start"
$ns_ at 4.0 "$cbr0 stop"

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 interval_ .07
$ns_ at 0.1 "$cbr1 start"
$ns_ at 4.1 "$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 interval_ .07
$ns_ at 2.4 "$cbr2 start"
$ns_ at 4.1 "$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 interval_ 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 interval_ 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 interval_ 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 interval_ 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 interval_ 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 interval_ 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 interval_ 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 interval_ 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 interval_ 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 interval_ 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
        $ns_ flush-trace
        close $tracefd
        exec nam datacache.nam &
        exit 0

    }

puts "Starting Simulation....."
$ns_ at 25.0 "stop"
$ns_ run
```

Procedure to run the program in the terminal window - $ns program9.tcl

OUTPUT



6.10 PROGRAM 10 – Creation of TCP (Transmission Control Protocol) communication between the nodes using DSDV routing protocol TCL script

Program Discription

Number of nodes in the network is static and is declared as three in the network. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Movement for each and every node is built with static speed and spectifed receiver address which is randomly generated and also the mobility will get change accoding to the time period.

Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is DSDV and the stop time of the simulation is 150ms. Three nodes are created which are node0, node 1 and node 2. Send TCP agent is created and attached to node0, destination TCPsink agent is created and attached to node1. Then the TCP agent and the TCPsink agent are connected. In the next level, FTP application is created and attached to the sender TCP agent. Now the communication is initiated.

File Name – program10.tcl

- Channel Type – Wireless Channel
- Propogation – Two Ray Ground Model
- Queue Type – DropTail
- Antenna Type – Omni Directional Antenna
- Number of nodes – 3
- Routing protocol - DSDV
- X dimension – 500
- Y dimension – 400
- Stop time – 150ms
- AGENT – TCP
- Application - FTP

```
# Define setting option
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)       3                          ;# 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)     150                        ;# time of simulation end
```

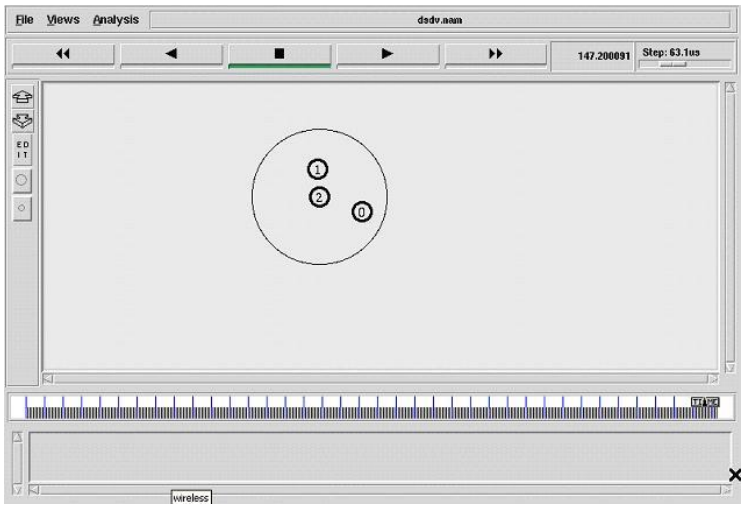
```
#Creating trace file and nam file
set tracefd       [open dsdv.tr w]
set windowVsTime2 [open win.tr w]
set namtrace      [open dsdv.nam w]

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

# set up topography object
set topo       [new Topography]

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

create-god $val(nn)

# 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_ 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"

# 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
$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
exec nam dsdv.nam &
exit 0
}

$ns run
```

Procedure to run the program in the terminal window - $ns program10.tcl

OUTPUT



6.11 PROGRAM 11 – Mobile node energy model construction TCL script

Program Discription

Number of nodes in the network is static and is declared as six in the network. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero.

Movement for each and every node is built with static speed and spectifed receiver address which is randomly generated and also the mobility will get change accoding to the time period. Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is DSR and the stop time of the simulation is 18ms.

File Name – program11.tcl

- Channel – Wireless Channel
- Propagation – Two Ray Ground Propagation
- Queue Type – Drop Tail
- Antenna – Omni Directional Antenna
- Initial Energy – 20 J
- Transmission Power – 0.9 J
- Receiver Power – 0.8 J
- Idle Power – 0.0 J
- Sense Power – 0.0175 J
- Routing Protocol – DSR
- Simulation Time – 18ms
- Number of nodes – 6 nodes
- X dimension – 750
- Y dimension – 550
- Initial Node Position - 30

```
## Setting The wireless Channels..

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)       5                          ;# max packet in ifq
set val(nn)           6                          ;# number of mobilenodes
set val(rp)           DSR                        ;# routing protocol
set val(x)            750                        ;# X dimension of topography
set val(y)            550                        ;# Y dimension of topography
set val(stop)         18.0                       ;# time of simulation end

## Create a simulator object(nothing but, a scheduler's object)..
    set ns [new Simulator]

## Create a trace file and nam file..
    set tracefd [open wireless1.tr w]
    set namtrace [open wireless1.nam w]

## Trace the nam and trace details from the main simulation..
    $ns trace-all $tracefd
    $ns namtrace-all-wireless $namtrace $val(x) $val(y)

## set up topography object..
    set topo [new Topography]

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

    set god_ [create-god $val(nn)]
```

```
## Color Descriptions..
    $ns color 1 dodgerblue
    $ns color 2 blue
    $ns color 3 cyan
    $ns color 4 green
    $ns color 5 yellow
    $ns color 6 black
    $ns color 7 magenta
    $ns color 8 gold
    $ns color 9 red

# Setting The Distance Variables..
# 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)
```

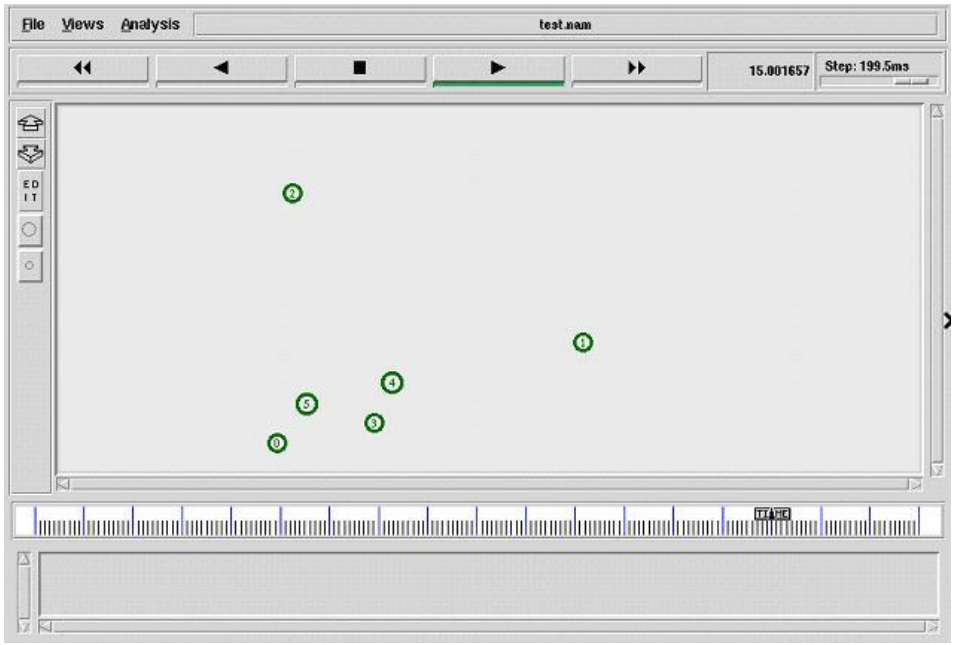
```
## Setting node config event with set of inputs..
    puts "Node Configuration Started here...\n \
            -channel $val(chan) \n \
            -adhocRouting $val(rp) \n \
            -llType $val(ll) \n \
            -macType $val(mac) \n \
            -ifqType $val(ifq) \n \
            -ifqLen $val(ifqlen) \n \
            -antType $val(ant) \n \
            -propType $val(prop) \n \
            -phyType $val(netif) \n"


     $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
# Energy model
    $ns node-config  -energyModel EnergyModel \
                -initialEnergy 20 \
                -txPower 0.9 \
                -rxPower 0.8 \
                -idlePower 0.0 \
                -sensePower 0.0175
```

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

for {set i 0} {$i < $val(nn)} {incr i} {
    $node_($i) color darkgreen
    $ns at 0.0 "$node_($i) color darkgreen"
    }

## Provide initial location of mobilenodes..

    if {$val(nn) >0} {
        for {set i 1} {$i < $val(nn) } { incr i } {
            set xx [expr rand()*600]
            set yy [expr rand()*500];
            $node_($i) set X_ $xx
            $node_($i) set Y_ $yy
        }
    }

## set god distance..
    $god_ set-dist 0 1 2
    $god_ set-dist 0 2 2
    $god_ set-dist 0 3 2
    $god_ set-dist 0 4 1
    $god_ set-dist 0 5 2
    $god_ set-dist 1 2 3
    $god_ set-dist 1 3 3


## 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 16.01 "puts \"end simulation\" " ;# $ns halt

 ## Stop procedure..

proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace

    exec nam wireless1.nam &
    exec 0
}

$ns run
```

Procedure to run the program in the terminal window - $ns program11.tcl

OUTPUT



6.12 PROGRAM 12 – Creation of nodes at random destination at particular time interval using AODV routing protocol TCL script

Program Discription

Number of nodes in the network is not static and is declared as six in the network. Nodes are configured in the mobile wireless node format. Number of nodes construction is given during the run time of the program. The user should give the number of nodes in the terminal window during the execution of the program. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Movement for each and every node is built with static speed and spectifed receiver address which is randomly generated and also the detination location will get change accoding to the time period.

File Name – program12.tcl

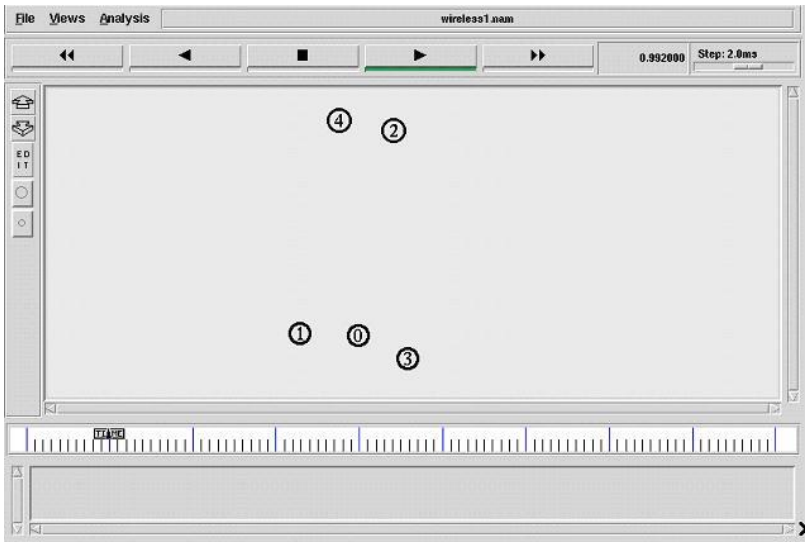
```
# Define options
set val(chan)         Channel/WirelessChannel    ;# channel type
set val(prop)         Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)        Phy/WirelessPhy            ;# network interface type
set val(mac)          Mac/802_11                 ;# MAC type
set 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)           5                          ;# number of mobilenodes
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)         10                         ;# time of simulation end

set ns            [new Simulator]

#Creating nam and trace file:

set tracefd       [open wireless1.tr w]
set namtrace      [open wireless1.nam w]

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

# set up topography object
set topo      [new Topography]

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

set god_ [create-god $val(nn)]
```

```
# 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

      ## Creating node objects..
for {set i 0} {$i < $val(nn) } { incr i } {
            set node_($i) [$ns node]
      }
      for {set i 0} {$i < $val(nn)  } {incr i } {
            $node_($i) color black
            $ns at 0.0 "$node_($i) color black"
      }

 ## Provide initial location of mobilenodes..
            for {set i 0} {$i < $val(nn) } { incr i } {
                  set xx [expr rand()*500]
                  set yy [expr rand()*400]
                  $node_($i) set X_ $xx
                  $node_($i) set Y_ $yy

            }
```

```
# 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";
}

#Destination procedure..
$ns at 0.0 "destination"
proc destination {} {
     global ns val node_
     set time 1.0
     set now [$ns now]
     for {set i 0} {$i<$val(nn)} {incr i} {
          set xx [expr rand()*500]
          set yy [expr rand()*400]
          $ns at $now "$node_($i) setdest $xx $yy 10.0"
     }
     $ns at [expr $now+$time] "destination"
}

$ns at $val(stop) "stop"

#Stop procedure

proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec nam wireless1.nam &
}

$ns run
```

Procedure to run the program in the terminal window - $ns program12.tcl

OUTPUT



6.13 PROGRAM 13 – Creation of nodes destination and random coloring using AODV routing protocol TCL script

Program Discription

Number of nodes in the network is not static and is declared as eight in the network. Nodes are configured in the mobile wireless node format. Number of nodes construction is given during the run time of the program. The user should give the number of nodes in the terminal window during the execution of the program. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Movement for each and every node is built with static speed and spectifed receiver address which is randomly generated and also the detination location will get change accoding to the time period. Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is AODV and the stop time of the simulation is 10ms. Here each and every group of the nodes is constructed with different type of colors.

File Name – program13.tcl

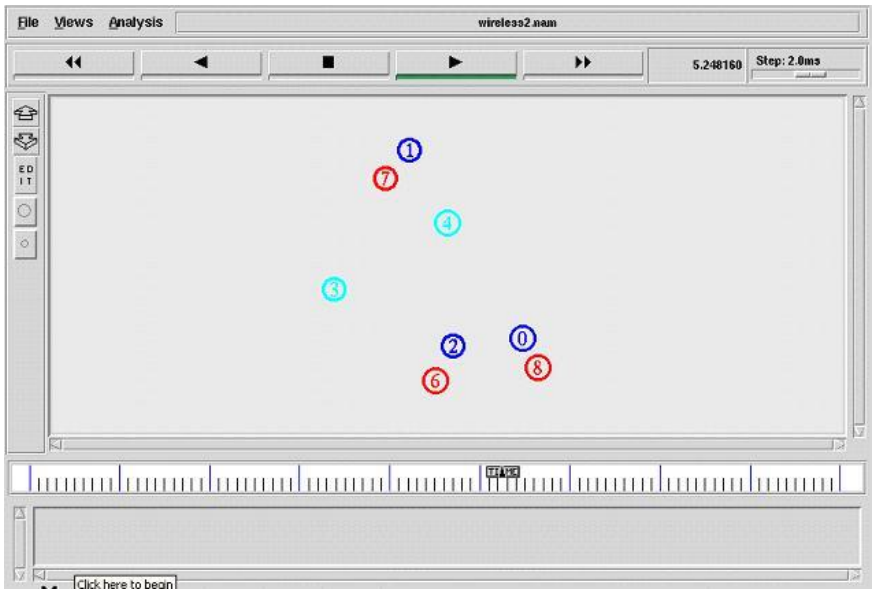
```
# Define setting options
set val(chan)         Channel/WirelessChannel    ;# channel type
set val(prop)         Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)        Phy/WirelessPhy            ;# network interface type
set val(mac)          Mac/802_11                 ;# MAC 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)           8                          ;# number of mobilenodes
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)         10                         ;# time of simulation end

set ns            [new Simulator]

#Creating nam and trace file:
set tracefd       [open wireless2.tr w]
set namtrace      [open wireless2.nam w]

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

# set up topography object
set topo       [new Topography]

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

set god_ [create-god $val(nn)]

# 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
```

```
## Creating node objects..
for{set i 0} {$i < 3 } { incr i } {
        set node_($i) [$ns node]
    }
    for{set i 0}{$i < 3  } {incr i } {
        $node_($i) color blue
        $ns at 0.0 "$node_($i) color blue"
    }
for{set i 3} {$i < 6 } { incr i } {
        set node_($i) [$ns node]
    }
    for{set i 3}{$i < 5  } {incr i } {
        $node_($i) color cyan
        $ns at 0.0 "$node_($i) color cyan"
    }
for{set i 5} {$i < 8 } { incr i } {
        set node_($i) [$ns node]
    }
    for{set i 5}{$i < 8  } {incr i } {
        $node_($i) color red
        $ns at 0.0 "$node_($i) color red"
    }

## Provide initial location of mobilenodes..
        for {set i 0} {$i < $val(nn) } { incr i } {
            set xx [expr rand()*500]
            set yy [expr rand()*400]
            $node_($i) set X_ $xx
            $node_($i) set Y_ $yy

        }

# 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";
}

# dynamic destination setting procedure..
$ns at 0.0 "destination"
```

```
proc destination {} {
    global ns val node_
    set time 1.0
    set now [$ns now]
    for {set i 0} {$i<$val(nn)} {incr i} {
        set xx [expr rand()*500]
        set yy [expr rand()*400]
        $ns at $now "$node_($i) setdest $xx $yy 10.0"
    }
    $ns at [expr $now+$time] "destination"
}

#stop procedure..
$ns at $val(stop) "stop"
proc stop {} {
   global ns tracefd namtrace
   $ns flush-trace
   close $tracefd
   close $namtrace
exec nam wireless2.nam &
}

$ns run
```

Procedure to run the program in the terminal window - $ns program13.tcl

OUTPUT



6.14 PROGRAM 14 – Creation of nodes with the initial and destination position in random manner using AODV routing protocol TCL script

Program Discription

Number of nodes in the network is not static and is declared as eight in the network. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero.

Movement for each and every node is built with static speed and spectifed receiver address which is randomly generated and also the detination location will get change accoding to the time period. Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is AODV and the stop time of the simulation is 10ms. Here each and every group of the nodes is constructed with different type of colors.

File Name – program14.tcl

- Channel – Wireless Channel
- Propagation – Two Ray Ground Propagation
- Queue Type – Drop Tail
- Antenna – Omni Directional Antenna
- Routing Protocol –AODV
- Simulation Time – 18ms
- Number of nodes – 8 nodes
- X dimension – 500
- Y dimension – 400
- Initial Node Position - 30

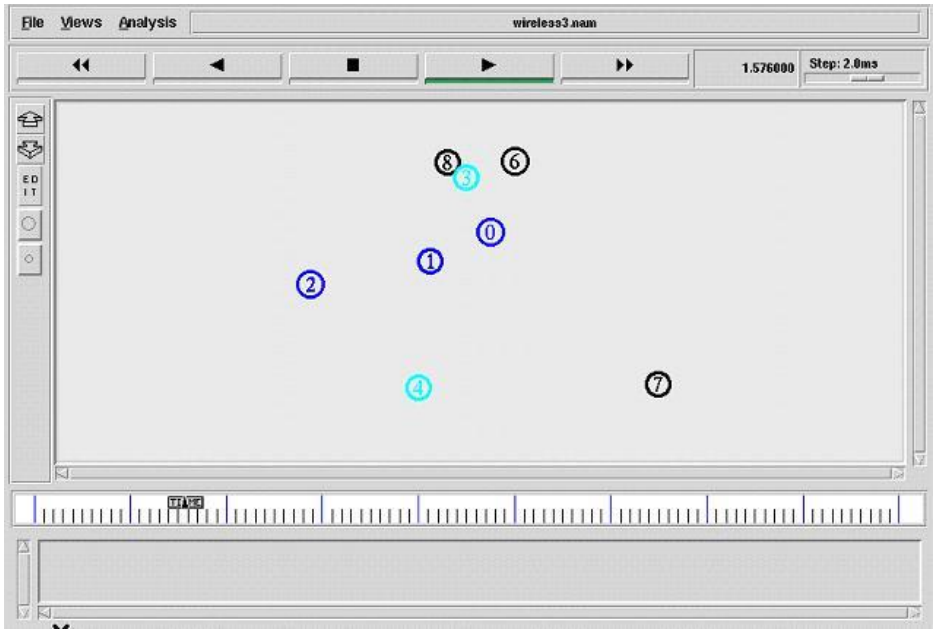
```
# Define options
set val(chan)         Channel/WirelessChannel          ;# channel type
set val(prop)         Propagation/TwoRayGround         ;# radio-propagation model
set val(netif)        Phy/WirelessPhy                  ;# network interface type
set val(mac)          Mac/802_11                       ;# 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)           8                                ;# number of mobilenodes
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)         10                               ;# time of simulation end

#Creating simulation:
set ns          [new Simulator]

#Creating nam and trace file:
set tracefd     [open wireless3.tr w]
set namtrace    [open wireless3.nam w]

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

# set up topography object
set topo      [new Topography]

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

set god_ [create-god $val(nn)]

# 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

## Creating node objects..
for {set i 0} {$i < 3 } { incr i } {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < 3  } {incr i } {
        $node_($i) color blue
        $ns at 0.0 "$node_($i) color blue"
    }
for {set i 3} {$i < 6 } { incr i } {
        set node_($i) [$ns node]
    }
    for {set i 3} {$i < 6  } {incr i } {
        $node_($i) color cyan
        $ns at 1.0 "$node_($i) color cyan"
    }
for {set i 6} {$i < 8 } { incr i } {
        set node_($i) [$ns node]
    }
    for {set i 5} {$i < 8  } {incr i } {
        $node_($i) color red
        $ns at 2.0 "$node_($i) color red"
    }

## Provide initial location of mobilenodes..
        for {set i 0} {$i < $val(nn) } { incr i } {
            set xx [expr rand()*500]
            set yy [expr rand()*400]
            $node_($i) set X_ $xx
            $node_($i) set Y_ $yy

        }

# 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";
}
```

```
# dynamic destination setting procedure..
$ns at 0.0 "destination"
proc destination {} {
     global ns val node_
     set time 1.0
     set now [$ns now]
     for {set i 0} {$i<$val(nn)} {incr i} {
          set xx [expr rand()*500]
          set yy [expr rand()*400]
          $ns at $now "$node_($i) setdest $xx $yy 10.0"
     }
     $ns at [expr $now+$time] "destination"
}

#stop procedure..
$ns at $val(stop) "stop"
proc stop {} {
   global ns tracefd namtrace
   $ns flush-trace
   close $tracefd
   close $namtrace
exec nam wireless3.nam &
}

$ns run
```

Procedure to run the program in the terminal window - $ns program14.tcl

OUTPUT



6.15 PROGRAM 15 – Creation of graphs with X dimension and Y Dimension constructed randomly using AODV routing protocol TCL script

Program Discription

Number of nodes in the network is static and is declared as three in the network. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Graph is randomly generated using the X and Y dimensions and is programmed to generate the trace file accordingly. Here the trace file acts as a input file to plot the graph in the format of trace file. Routing protocol is AODV and the stop time of the simulation is 10ms.

File Name – program15.tcl

- Channel – Wireless Channel
- Propagation – Two Ray Ground Propagation
- Queue Type – Drop Tail
- Antenna – Omni Directional Antenna
- Routing Protocol –AODV
- Simulation Time – 10ms
- Initial Node Position - 30

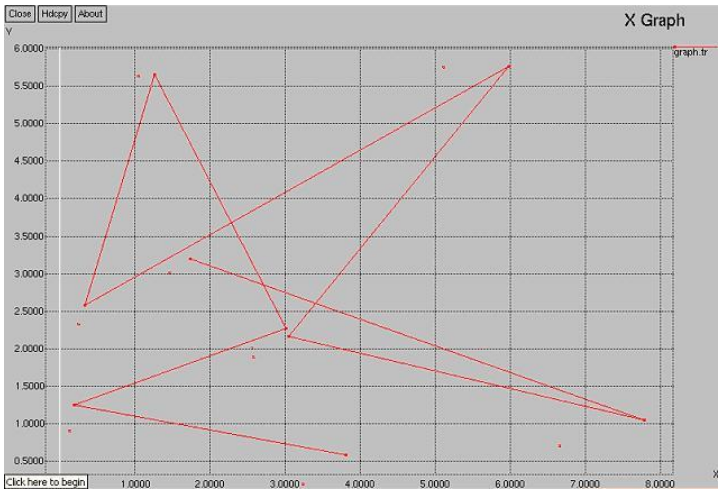
```
# Define setting options
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/802_11                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# 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)     3                          ;# number of mobilenodes
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)   10                         ;# time of simulation end

set ns          [new Simulator]
```

```
# Creating simulation
set ns [new Simulator]

#Creating nam and trace file
set tracefd      [open Graph1.tr w]
set namtrace     [open Graph1.nam w]

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

# set up topography object
set topo      [new Topography]

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

set god_ [create-god $val(nn)]

# 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

## Creating node objects..
for {set i 0} {$i < $val(nn) } { incr i } {
         set node_($i) [$ns node]
    }
    for {set i 0} {$i < $val(nn)  } {incr i } {
         $node_($i) color black
         $ns at 0.0 "$node_($i) color black"
    }

# Provide initial location of mobilenodes
$node_(0) set X_ 50.0
$node_(0) set Y_ 50.0
$node_(0) set Z_ 0.0
```

```
$node_(1) set X_ 200.0
$node_(1) set Y_ 250.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 300.0
$node_(2) set Y_ 300.0
$node_(2) set Z_ 0.0

# 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 10.01 "puts \"end simulation\" ; $ns halt"
$ns at 1.0 "Graph"
set g [open graph.tr w]
proc Graph {} {
global ns g
set time 1.0
set now [$ns now]
puts $g "[expr rand()*8] [expr rand()*6]"

$ns at [expr $now+$time] "Graph"
}

#Stop proceture
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec xgraph -M -bb -geometry 700X800 graph.tr &
exec nam Graph1.nam &
exit 0
}

$ns run
```

Procedure to run the program in the terminal window - $ns program15.tcl

OUTPUT



6.16 PROGRAM 16 – Creation of graphs with two parameters as inputs using AODV routing protocol TCL script

Program Discription

Number of nodes in the network is static and is declared as three in the network. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Graph is randomly generated using the X and Y dimensions and is programmed to generate the trace file accordingly. The trace file acts as input file to plot the graph in the format of trace file. Here single plotted graph consist of two trace file values. Different colors are given to each trace file during plotting. Routing protocol is AODV and the stop time of the simulation is 10ms.

File Name – program16.tcl

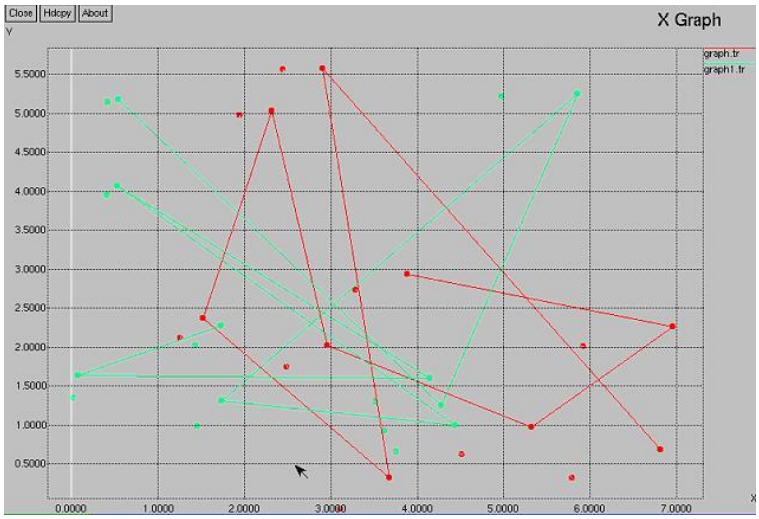
```
# Define setting options
set val(chan)    Channel/WirelessChannel     ;# channel type
set val(prop)    Propagation/TwoRayGround    ;# radio-propagation model
set val(netif)   Phy/WirelessPhy             ;# network interface type
set val(mac)     Mac/802_11                  ;# MAC type
set val(ifq)     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)      3                           ;# number of mobilenodes
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)    10                          ;# time of simulation end

set ns           [new Simulator]
```

```
#Creating simulation
set ns [new Simulator]

#Creating nam and trace file
set tracefd      [open Graph2.tr w]
set namtrace     [open Graph2.nam w]

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

# set up topography object
set topo      [new Topography]

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

set god_ [create-god $val(nn)]

# 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

## Creating node objects..
for {set i 0} {$i < $val(nn) } { incr i } {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < $val(nn) } {incr i} {
        $node_($i) color black
        $ns at 0.0 "$node_($i) color black"
    }
```

```
# Provide initial location of mobilenodes
$node_(0) set X_ 50.0
$node_(0) set Y_ 50.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 200.0
$node_(1) set Y_ 250.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 300.0
$node_(2) set Y_ 300.0
$node_(2) set Z_ 0.0

# 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 10.01  "puts \"end simulation\" ; $ns halt"

#Graph procedure..
#procedure..
$ns at 1.0 "Graph"
set g [open graph.tr w]
set g1 [open graph1.tr w]
proc Graph {} {
global ns g g1
set time 1.0
set now [$ns now]
puts $g "[expr rand()*8] [expr rand()*6]"
puts $g1 "[expr rand()*8] [expr rand()*6]"
$ns at [expr $now+$time] "Graph"
}


#stop procedure:
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec xgraph -P -bb -geometry 700X800 graph.tr graph1.tr &

exec nam  Graph2.nam  &
exit 0
}

$ns run
```

Procedure to run the program in the terminal window - $ns program16.tcl

OUTPUT



6.17 PROGRAM 17 – Creation of graphs with more than two parameter files as inputs using AODV routing protocol TCL script

Program Discription

Number of nodes in the network is static and is declared as three in the network. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Graph is randomly generated using the X and Y dimensions and is programmed to generate the trace file accordingly. The trace file acts as input file to plot the graph in the format of trace file. Here single plotted graph consist of more than two trace file values. Different colors are given to each trace file during plotting. Routing protocol is AODV and the stop time of the simulation is 10ms.

File Name – program17.tcl

```
# Define setting 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)      3                           ;# number of mobilenodes
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)    10                          ;# time of simulation end
```

```
# Creating simulation
set ns [new Simulator]

#Creating nam and trace file
set tracefd     [open Graph3.trw]
set namtrace    [open Graph3.nam w]

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

# set up topography object
set topo     [new Topography]

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

set god_ [create-god $val(nn)]



# 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

# Creating node objects..
for {set i 0} {$i < $val(nn)} {incr i} {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < $val(nn) } {incr i} {
        $node_($i) color black
        $ns at 0.0 "$node_($i) color black"
    }
```

```
# Provide initial location of mobilenodes
$node_(0) set X_ 50.0
$node_(0) set Y_ 50.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 200.0
$node_(1) set Y_ 250.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 300.0
$node_(2) set Y_ 300.0
$node_(2) set Z_ 0.0

# 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 10.01 "puts \"end simulation\" ; $ns halt"

#Graph procedure..
$ns at 1.0 "Graph"
set g [open graph.trw]
set g1 [open graph1.trw]
set g2 [open graph2.trw]
proc Graph {} {
global ns g g1
set time 1.0
set now [$ns now]
puts $g "[expr rand()*8] [expr rand()*6]"
puts $g1 "[expr rand()*8] [expr rand()*6]"
puts $g2 "[expr rand()*8] [expr rand()*6]"
$ns at [expr $now+$time] "Graph"
}
```

```
#Stop proceture
proc stop {} {
   global ns tracefd namtrace
   $ns flush-trace
   close $tracefd
   close $namtrace
exec xgraph -M -bb -geometry 700X800 graph.tr graph1.tr graph2.tr &
exec nam Graph3.nam &
exit 0
}

$ns run
```

Procedure to run the program in the terminal window - $ns program17.tcl

OUTPUT