EX.NO: 9	SIMULATION OF DISTANCE VECTOR AND LINK STATE ROUTING ALGORITHM
DATE:	

AIM:

To simulate the Distance vector routing and Link state routing protocols using NS2.

DISTANCE VECTOR:

ALGORITHM:

1. Create a simulator object
2. Set routing protocol to Distance Vector routing
3. Trace packets on all links onto NAM trace and text trace file
4. Define finish procedure to close files, flush tracing and run NAM
5. Create eight nodes
6. Specify the link characteristics between nodes
7. Describe their layout topology as a octagon
8. Add UDP agent for node n1
9. Create CBR traffic on top of UDP and set traffic parameters.
10. Add a sink agent to node n4
11. Connect source and the sink
12. Schedule events as follows:
    a. Start traffic flow at 0.5
    b. Down the link n3-n4 at 1.0
    c. Up the link n3-n4 at 2.0
    d. Stop traffic at 3.0
    e. Call finish procedure at 5.0
13. Start the scheduler
14. Observe the traffic route when link is up and down
15. View the simulated events and trace file analyze it
16. Stop

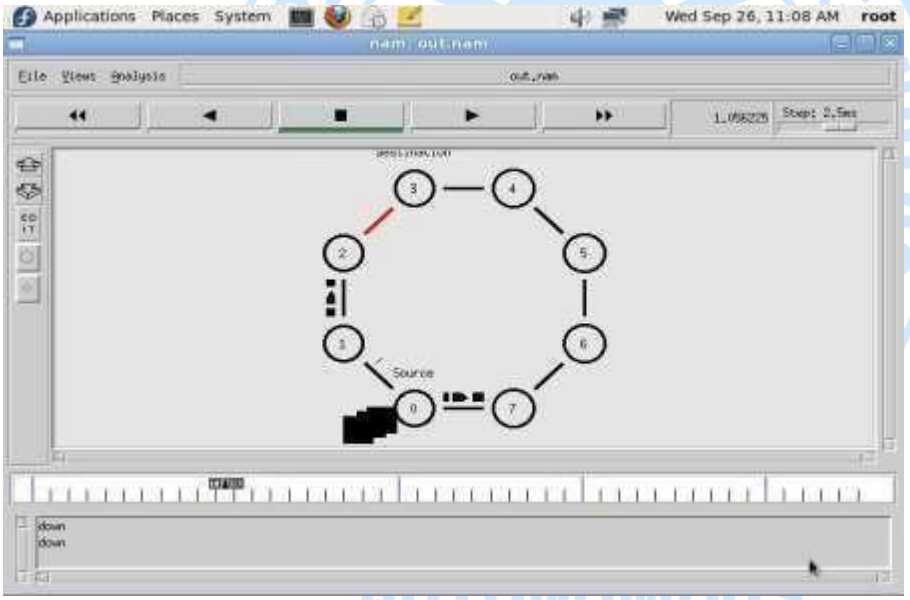
PROGRAM:

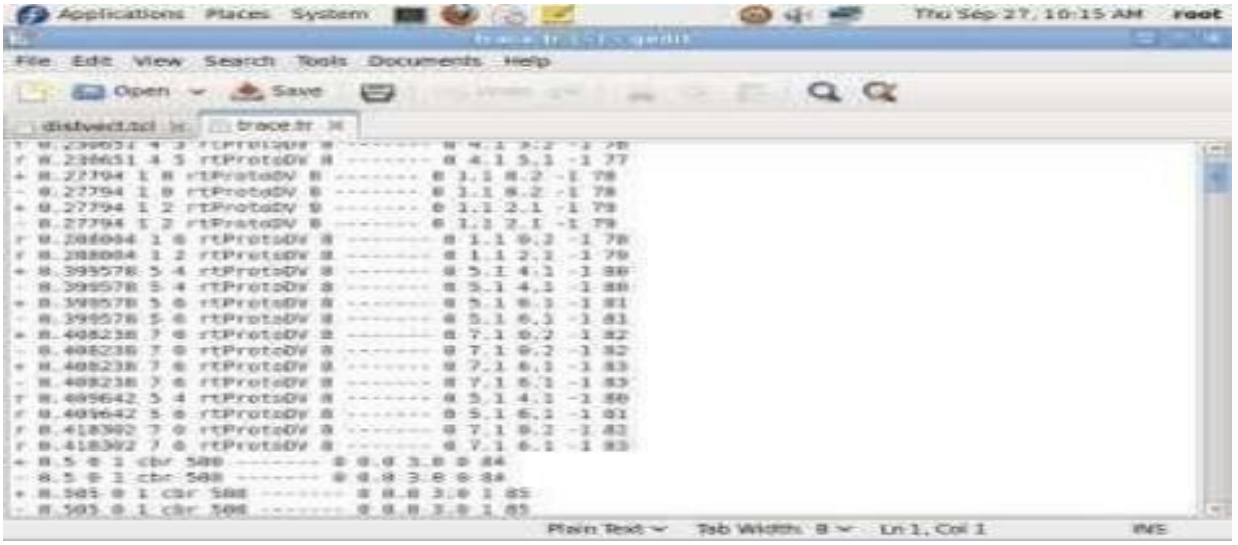
```
#Distance vector routing protocol – distvect.tcl
#Create a simulator object
set ns [new Simulator]
#Use distance vector routing
$ns rtproto DV
#Open the nam trace file set nf [open out.nam w]
$ns namtrace-all $nf # Open tracefile
```

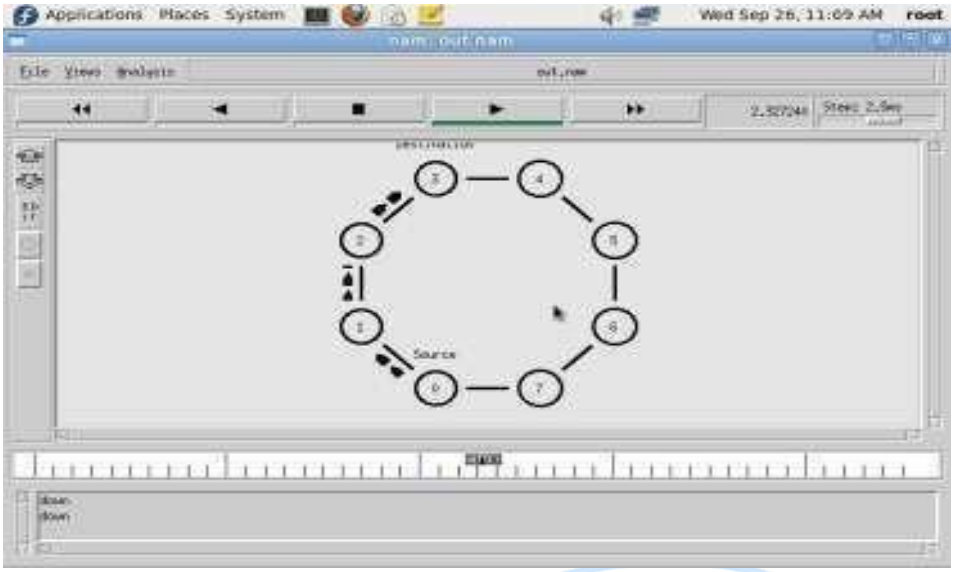
```
set nt [open trace.tr w]
$ns trace-all $nt
#Define 'finish' procedure
proc finish {}
{     global ns nf
      $ns flush-trace
      #Close the trace file
      close $nf
      #Execute nam on the trace file
      exec nam -a out.nam &
      exit 0
}
#  Create  8  nodes
set  n1  [$ns  node]
set  n2  [$ns  node]
set  n3  [$ns  node]
set  n4  [$ns  node]
set  n5  [$ns  node]
set  n6  [$ns  node]
set  n7  [$ns  node]
set n8 [$ns node]
# Specify link characterestics
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link $n4 $n5 1Mb 10ms DropTail
$ns duplex-link $n5 $n6 1Mb 10ms DropTail
$ns duplex-link $n6 $n7 1Mb 10ms DropTail
$ns duplex-link $n7 $n8 1Mb 10ms DropTail
$ns duplex-link $n8 $n1 1Mb 10ms DropTail
$ns duplex-link-op $n1 $n2 orient left-up
$ns duplex-link-op $n2 $n3 orient up
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n4 $n5 orient right
$ns duplex-link-op $n5 $n6 orient right-down
$ns duplex-link-op $n6 $n7 orient down
$ns duplex-link-op $n7 $n8 orient left-down
$ns duplex-link-op $n8 $n1 orient left
#Create a UDP agent and attach it to node n1 set udp0 [new Agent/UDP]
$ns attach-agent $n1 $udp0
#Create a CBR traffic source and attach it to
udp0set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
```

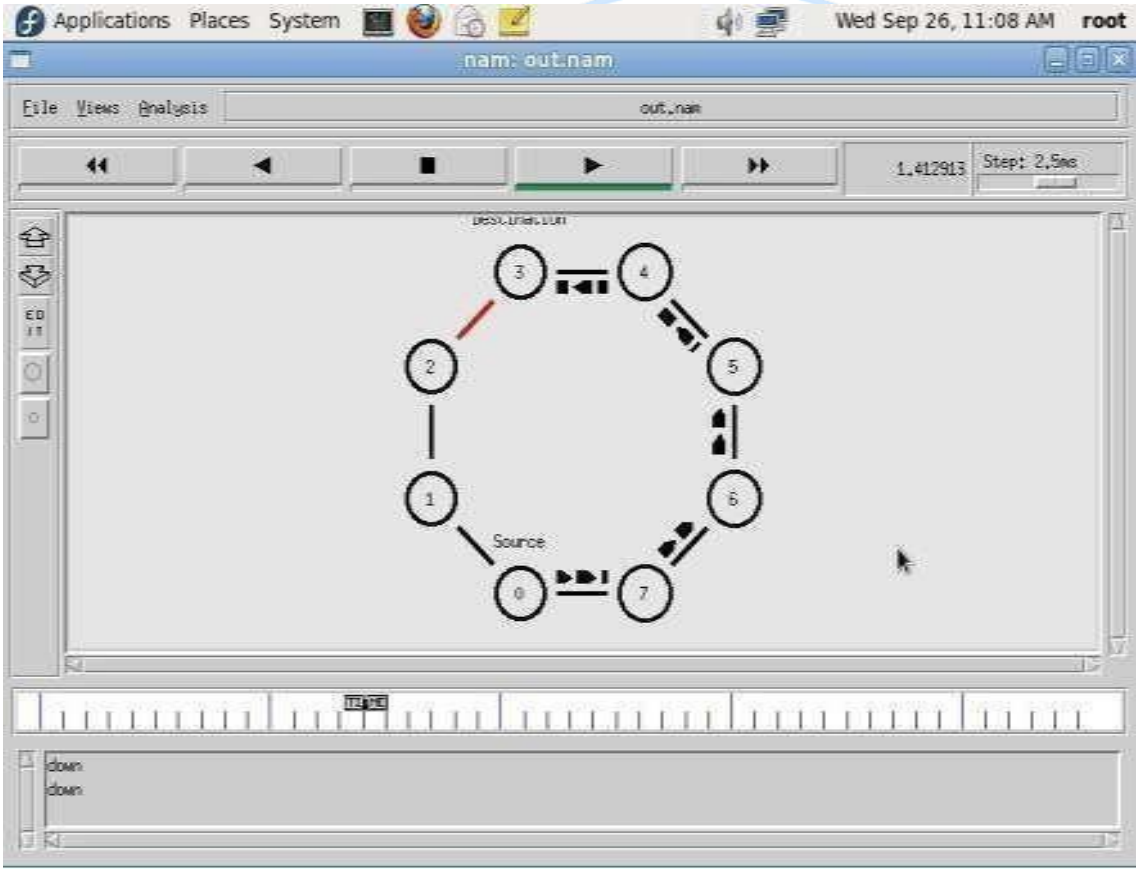
```
$cbr0 attach-agent $udp0
#Create a Null agent (a traffic sink) and attach it to node n4 set null0 [new Agent/Null]
$ns attach-agent $n4 $null0
#Connect the traffic source with the traffic sink
$ns connect $udp0 $null0
#Schedule events for the CBR agent and the network dynamics
$ns at 0.0 "$n1 label Source"
$ns at 0.0 "$n4 label Destination"
$ns at 0.5 "$cbr0 start"
$ns rtmodel-at 1.0 down $n3 $n4
$ns rtmodel-at 2.0 up $n3 $n4
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish" #Run the simulation
$ns run
```

Output:









LINK STATE ROUTING:
ALGORITHM:

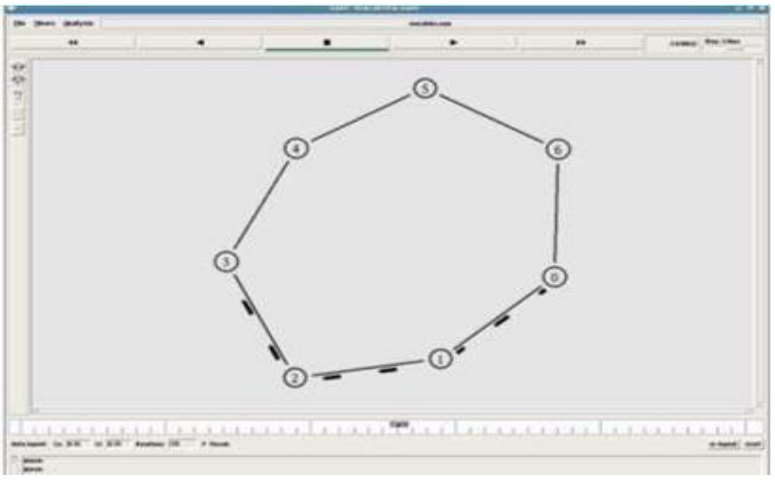
1. Create a Simulator object.
2. Set routing as dynamic.
3. Open the trace and nam trace files.
4. Define the finish procedure.
5. Create nodes and the links between them.
6. Create the agents and attach them to the nodes.
7. Create the applications and attach them to the udp agent.
8. Connect udp and null..

9. At 1 sec the link between node 1 and 2 is broken.
10. At 2 sec the link is up again.
11. Run the simulation.

PROGRAM:

```
set ns [new Simulator]
$ns rtproto LS
set nf [open linkstate.nam w]
$ns namtrace-all $nf
set f0 [open linkstate.tr w]
$ns trace-all $f0 proc finish {} {
    global ns f0 nf
    $ns flush-trace
    close $f0 close
    $nf
    exec nam linkstate.nam & exit 0
for {set i 0} {$i <7} {incr i}
    { set n($i) [$ns node]
}
for {set i 0} {$i <7} {incr i} {
    $ns duplex-link $n($i) $n([expr ($i+1)%7]) 1Mb 10ms DropTail
}
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0 set null0 [new Agent/Null]
$ns attach-agent $n(3) $null0
$ns connect $udp0 $null0
$ns at 0.5 "$cbr0 start"
$ns rtmodel-at 1.0 down $n(1) $n(2)
$ns rtmodel-at 2.0 up $n(1) $n(2)
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run
```

OUTPUT:



RESULT:

Thus the simulation for Distance vector routing and the Link state routing protocols was done using NS2.

EX.NO:8	Study of TCP/UDP performance using Simulation tool
DATE:	

AIM:

To simulate the performance of TCP/UDP using NS2.

TCP PERFORMANCE
ALGORITHM:

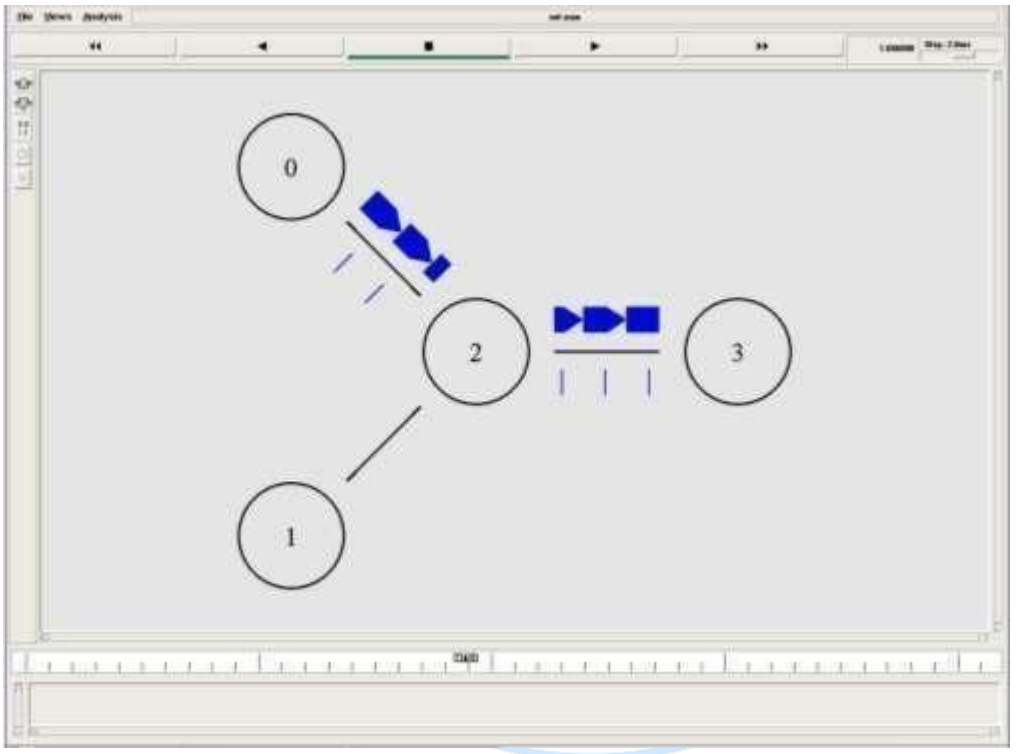
1. Create a Simulator object.
2. Set routing as dynamic.
3. Open the trace and nam trace files.
4. Define the finish procedure.
5. Create nodes and the links between them.
6. Create the agents and attach them to the nodes.
7. Create the applications and attach them to the tcp agent.
8. Connect tcp and tcp sink.
9. Run the simulation.

PROGRAM:

```
set ns [new Simulator]
$ns color 0 Blue
$ns color 1 Red
$ns color 2 Yellow set n0 [$ns node] set n1 [$ns node] set n2 [$ns node] set n3 [$ns
node]set f [open tcpout.tr w]
$ns trace-all $f
set nf [open tcpout.nam w]
$ns namtrace-all $nf
$ns duplex-link $n0 $n2 5Mb 2ms DropTail
$ns duplex-link $n1 $n2 5Mb 2ms DropTail
$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail
$ns duplex-link-op $n0 $n2 orient right-up
$ns duplex-link-op $n1 $n2 orient right-down
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n2 $n3 queuePos 0.5 set tcp [new Agent/TCP]
$tcp set class_ 1
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $tcp
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
```

```
$ns at 1.2 "$ftp start"
$ns at 1.35 "$ns detach-agent $n1 $tcp ; $ns detach-agent $n3 $sink"
$ns at 3.0 "finish" proc finish {} {
    global ns f nf
    $ns flush-trace close $f
    close $nf
    puts "Running nam.."
    exec xgraph tcpout.tr -geometry 600x800 & exec nam tcpout.nam
    &exit 0
}
$ns run
```

OUTPUT:



UDP PERFORMANCE:

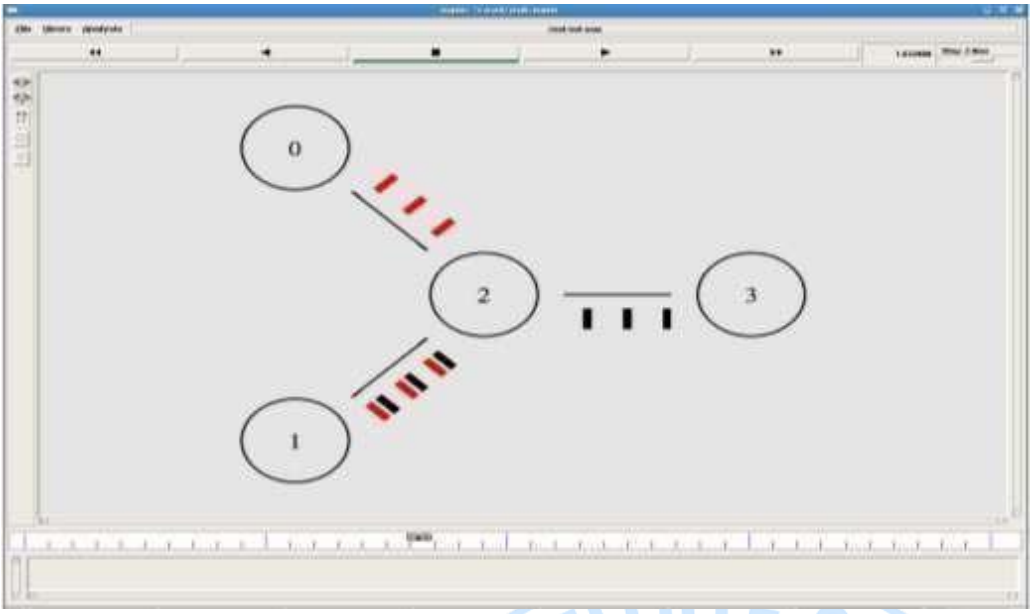
ALGORITHM:

1. Create a Simulator object.
2. Set routing as dynamic.
3. Open the trace and nam trace files.
4. Define the finish procedure.
5. Create nodes and the links between them.
6. Create the agents and attach them to the nodes.
7. Create the applications and attach them to the UDP agent.
8. Connect udp and null agents.
9. Run the simulation.

PROGRAM:

```
set ns [new Simulator]
$ns color 0 Blue
$ns color 1 Red
$ns color 2 Yellow set n0 [$ns node] set n1 [$ns node] set n2 [$ns node] set n3 [$ns
node]set f [open udpout.tr w]
$ns trace-all $f
set nf [open udpout.nam w]
$ns namtrace-all $nf
$ns duplex-link $n0 $n2 5Mb 2ms DropTail
$ns duplex-link $n1 $n2 5Mb 2ms DropTail
$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail
$ns duplex-link-op $n0 $n2 orient right-up
$ns duplex-link-op $n1 $n2 orient right-down
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n2 $n3 queuePos 0.5 set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0 set udp1 [new Agent/UDP]
$ns attach-agent $n3 $udp1
$udp1 set class_ 0
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1 set null0 [new Agent/Null]
$ns attach-agent $n1 $null0 set null1 [new Agent/Null]
$ns attach-agent $n1 $null1
$ns connect $udp0 $null0
$ns connect $udp1 $null1
$ns at 1.0 "$cbr0 start"
$ns at 1.1 "$cbr1 start"
puts [$cbr0 set packetSize_] puts [$cbr0 set interval_]
$ns at 3.0 "finish" proc finish {} {
    global ns f nf
    $ns flush-trace close $f
    close $nf
    puts "Running nam.." exec nam udpout.nam & exit 0
}
$ns run
```

OUTPUT:



RESULT :

Thus the study of TCP/UDP performance is done successfully.