Name: Soham Desai
XIE ID: 202003021
Roll no:11
Batch: B

Experiment 5

Aim: Implement distance vector and link state routing protocols in NS2

LO NO: 3,5

LO statement: Demonstrate and measure different network scenarios and their performance behavior.
Analyze the traffic flow of different protocols.

Theory:

Distance vector algorithm:

- The Distance vector algorithm is iterative, asynchronous and distributed.
- Distributed: It is distributed in that each node receives information from one or more of its directly attached neighbors, performs calculation and then distributes the result back to its neighbors.
- Iterative: It is iterative in that its process continues until no more information is available to be exchanged between neighbors.
- Asynchronous: It does not require that all of its nodes operate in the lock step with each other.
- The Distance vector algorithm is a dynamic algorithm.
- It is mainly used in ARPANET, and RIP.
- Each router maintains a distance table known as Vector.
- Link state routing is a technique in which each router shares the knowledge of its neighborhood with every other router in the internetwork.

The three keys to understand the Link State Routing algorithm:

- Knowledge about the neighborhood: Instead of sending its routing table, a router sends the information about its neighborhood only. A router broadcast its identities and cost of the directly attached links to other routers.
- Flooding: Each router sends the information to every other router on the internetwork except its neighbors. This process is known as Flooding. Every router that receives the packet sends the copies to all its neighbors. Finally, each and every router receives a copy of the same information.
- Information sharing: A router sends the information to every other router only when the change occurs in the information.
- The Link state routing algorithm is also known as Dijkstra's algorithm which is used to find the shortest path from one node to every other node in the network.

- The Dijkstra's algorithm is an iterative, and it has the property that after kth iteration of the algorithm, the least cost paths are well known for k destination nodes.

Code 1:

Code for distance vector routing protocol simulation:

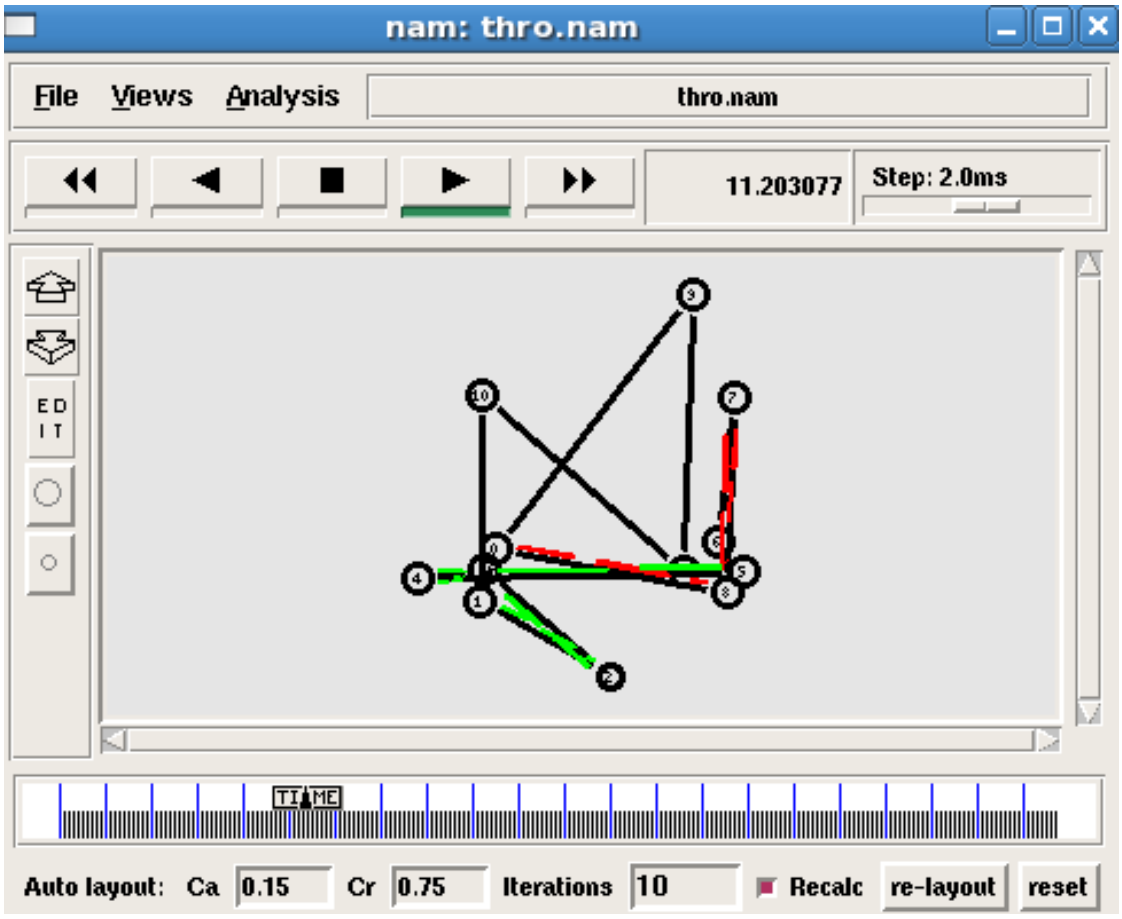
```
set ns [new Simulator]
set nr [open thro.tr w]
$ns trace-all $nr
set nf [open thro.nam w]
$ns namtrace-all $nf
proc finish {}
{ global ns nr
$ns flush-trace close $nf close $nr
exec nam thro.nam &amp;amp; exit 0
}
for { set i 0 } { $i &amp;lt; 12} { incr i 1 } { set n($i) [$ns node]}
for {set i 0} {$i &amp;lt; 8} {incr i} {
$ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail }
$ns duplex-link $n(0) $n(8) 1Mb 10ms DropTail
$ns duplex-link $n(1) $n(10) 1Mb 10ms DropTail
$ns duplex-link $n(0) $n(9) 1Mb 10ms DropTail
$ns duplex-link $n(9) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(10) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(11) $n(5) 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(5) $null0
$ns connect $udp0 $null0 set udp1 [new Agent/UDP]
$ns attach-agent $n(1) $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1 set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp1 $null0
```

Name: Soham Desai
XIE ID: 202003021
Roll no:11
Batch: B

```
$ns rtproto DV
$ns rtmodel-at 10.0 down $n(11) $n(5)
$ns rtmodel-at 15.0 down $n(7) $n(6)
$ns rtmodel-at 30.0 up $n(11) $n(5)
$ns rtmodel-at 20.0 up $n(7) $n(6)
$udp0 set fid_ 1
$udp1 set fid_ 2
$ns color 1 Red
$ns color 2 Green
$ns at 1.0 &amp;quot;$cbr0 start&amp;quot;
$ns at 2.0 &amp;quot;$cbr1 start&amp;quot;
$ns at 45 &amp;quot;finish&amp;quot;
$ns run
```

Output:



Name: Soham Desai
XIE ID: 202003021
Roll no:11
Batch: B

Code 2:

Code for Link state routing protocol simulation:

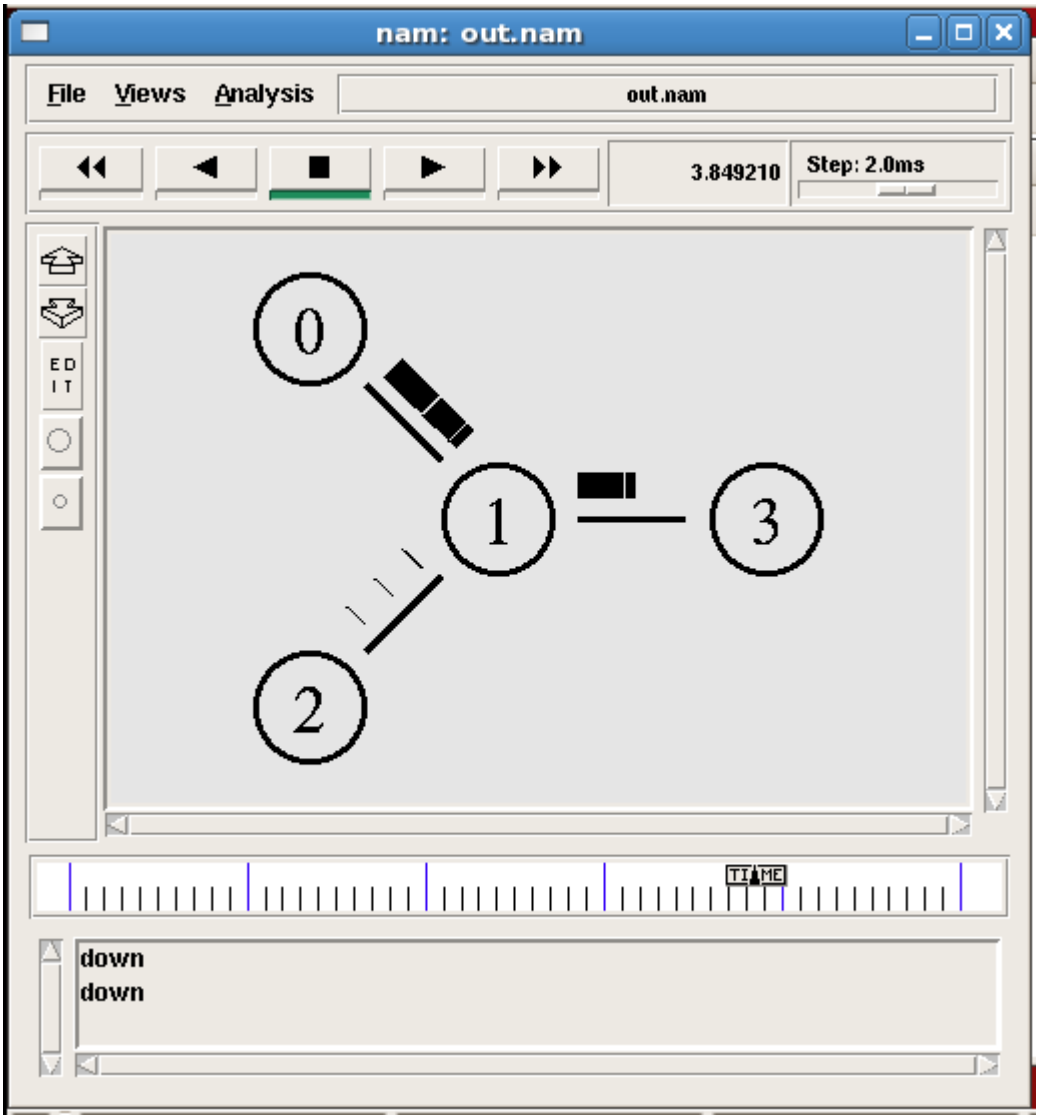
```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
set tr [open out.tr w]
$ns trace-all $tr
proc finish {} {
global nf ns tr
$ns flush-trace
close $tr
exec nam out.nam &
exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n0 $n1 10Mb 10ms DropTail
$ns duplex-link $n1 $n3 10Mb 10ms DropTail
$ns duplex-link $n2 $n1 10Mb 10ms DropTail
$ns duplex-link-op $n0 $n1 orient right-down
$ns duplex-link-op $n1 $n3 orient right
$ns duplex-link-op $n2 $n1 orient right-up
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set ftp [new Application/FTP]
$ftp attach-agent $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
set udp [new Agent/UDP]
$ns attach-agent $n2 $udp
set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $tcp $sink
```

Name: Soham Desai
XIE ID: 202003021
Roll no:11
Batch: B

```
$ns connect $udp $null
$ns rtmodel-at 1.0 down $n1 $n3
$ns rtmodel-at 2.0 up $n1 $n3
$ns rtproto LS
$ns at 0.0 "$ftp start"
$ns at 0.0 "$cbr start"
$ns at 5.0 "finish"
$ns run
```

Output 2:



Conclusion: From this experiment we can conclude that we can implement distance vector and link state routing protocol in NS2.