THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Name: Bhavya Sheth

Roll No.: 97

Date: 05-03-2021

Lab Assignment No.: 05

Aim: Simulation of Network with specific routing protocols (Distance Vector, Link State).

Lab Outcome Attained: LO 3: -To understand the network simulator environment and visualize a network topology and observe its performance. And LO 5: -To observe and study the traffic flow and the contents of protocol frames.

Theory:

The user level simulation script requires one command: to specify the unicast routing strategy or protocols for the simulation. A routing strategy is a general mechanism by which *ns* will compute routes for the simulation. There are four routing strategies in *ns*: Static, Session, Dynamic and Manual. Conversely, a routing protocol is a realization of a specific algorithm. Currently, Static and Session routing use the Dijkstra's all-pairs SPF algorithm

If a simulation script does not specify any []rtproto command, then *ns* will run Static routing on all the nodes in the topology.

Multiple []rtproto lines for the same or different routing protocols can occur in a simulation script. However, a simulation cannot use both centralized routing mechanisms such as static or session routing and detailed dynamic routing protocols such as DV.

In dynamic routing, each node can be running more than one routing protocol. In such situations, more than one routing protocol can have a route to the same destination. Therefore, each protocol affixes a preference value to each of its routes. These values are non-negative integers in the range 0...255. The lower the value, the more preferred the route. When multiple routing protocol agents have a route to the same destination, the most preferred route is chosen and installed in the node's forwarding tables. If more than one agent has the most preferred routes, the ones with the lowest metric is chosen. We call the least cost route from the most preferred protocol the ``candidate''

route. If there are multiple candidate routes from the same or different protocols, then, currently, one of the agent's routes is randomly chosen

A distance-vector routing protocol in data networks determines the best route for data packets based on distance. Distance-vector routing protocols measure the distance by the number of routers a packet has to pass, one router counts as one hop. Some distance-vector protocols also take into account network latency and other factors that influence traffic on a given route. To determine the best route across a network, routers, on which a distance-vector protocol is implemented, exchange information with one another, usually routing tables plus hop counts for destination networks and possibly other traffic information. Distance-vector routing protocols also require that a router informs its neighbours of network topology changes periodically.

Link state protocols are also called shortest-path-first protocols. Link state routing protocols have a complete picture of the network topology. Hence, they know more about the whole network than any distance vector protocol. Three separate tables are created on each link state routing enabled router. One table is used to hold details about directly connected neighbours, one is used to hold the topology of the entire internetwork and the last one is used to hold the actual routing table.
Link state protocols send information about directly connected links to all the routers in the network. Examples of Link state routing protocols include OSPF - Open Shortest Path First and IS-IS - Intermediate System to Intermediate System**.** There are also routing protocols that are considered to be hybrid in the sense that they use aspects of both distance vector and link state protocols. EIGRP - Enhanced Interior Gateway Routing Protocol is one of those hybrid routing protocols.

Executable Code:

```
set ns [new Simulator]

set nr [open bhavya.tr w]

$ns trace-all $nr

set nf [open bhavya.nam w]

$ns namtrace-all $nf
```

```
proc finish { } {
global ns nr nf
$ns flush-trace
close $nf
close $nr
exec nam bhavya.nam &
  exit 0
}

for { set i 0 } { $i < 10} { incr i 1 } {
set n($i) [$ns node]}

for {set i 0} {$i < 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(5) 1Mb 10ms DropTail
$ns duplex-link $n(0) $n(9) 1Mb 10ms DropTail
$ns duplex-link $n(9) $n(3) 1Mb 10ms DropTail
$ns duplex-link $n(1) $n(7) 1Mb 10ms DropTail
$ns duplex-link $n(4) $n(9) 1Mb 10ms DropTail
$ns duplex-link $n(5) $n(4) 1Mb 10ms DropTail
```

```
#Give node position (for NAM)

$ns duplex-link-op $n(0) $n(8) orient right

$ns duplex-link-op $n(1) $n(5) orient right-up

$ns duplex-link-op $n(0) $n(9) orient right

$ns duplex-link-op $n(3) $n(4) orient right-down

$ns duplex-link-op $n(9) $n(3) orient right-up

$ns duplex-link-op $n(1) $n(7) orient right

$ns duplex-link-op $n(4) $n(9) orient right-down

$ns duplex-link-op $n(5) $n(4) orient right


#Setup a TCP connection

set tcp [new Agent/TCP]

#$tcp set class_ 2

$ns attach-agent $n(1) $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n(5) $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


$ns at 1.0 "$ftp start"

$ns at 4.0 "$ftp stop"


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_ 1000

$cbr1 set  interval_ 0.010

$cbr1 attach-agent $udp1

set null0 [new Agent/Null]
```

```
$ns attach-agent $n(5) $null0

$ns connect $udp1 $null0


$ns rtproto DV


$ns rtmodel-at 10.0 down $n(1) $n(5)

$ns rtmodel-at 15.0 down $n(7) $n(6)

$ns rtmodel-at 30.0 up $n(1) $n(5)

$ns rtmodel-at 20.0 up $n(7) $n(6)


$udp0 set fid_ 1

$udp1 set fid_ 2


$ns color 1 orange

$ns color 2 red


$ns at 1.0 "$cbr0 start"

$ns at 2.0 "$cbr1 start"


$ns at 45 "finish"

$ns run
```

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Name: Bhavya Sheth

Roll No.: 97

Explanation of Code:

- for { set i 0 } { $i < 10} { incr i 1 } {

  set n($i) [$ns node]}

This set of code basically creates the nodes from 0 to 9. Note that arrays, just like other variables in Tcl, don't have to be declared first.

- for {set i 0} {$i < 8} {incr i} {

  $ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms Drop Tail }

This set of code connects the nodes to create a circular topology. This 'for' loop connects all nodes with the next node in the array with the exception of the last node, which is being connected with the first node. To accomplish that, I used the '%' (modulo) operator. When you run the script now, the topology might look a bit strange in nam at first, but after you hit the 're-layout' button it should look like the picture below.

- $ns rtproto DV

It uses dynamic routing. Start the simulation again, and you will see how at first a lot of small packets run through the network. If you slow nam down enough to click on one of them, you will see that they are 'rtProtoDV' packets which are being used to exchange routing information between the nodes. When the link goes down again at 1.0 seconds, the routing will be updated and the traffic will be re-routed through the nodes.

- $ns rtmodel-at 10.0 down $n(1) $n(5)

  $ns rtmodel-at 15.0 down $n(7) $n(6)

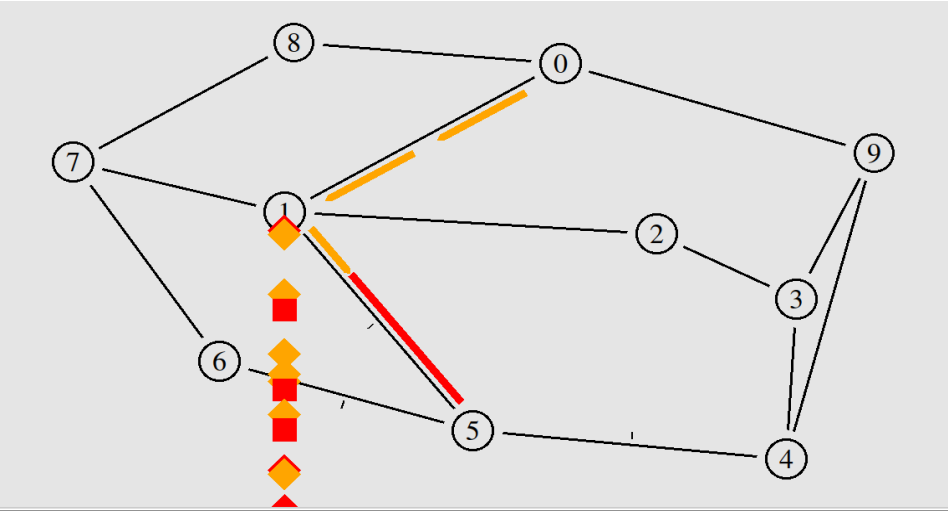
  $ns rtmodel-at 30.0 up $n(1) $n(5)

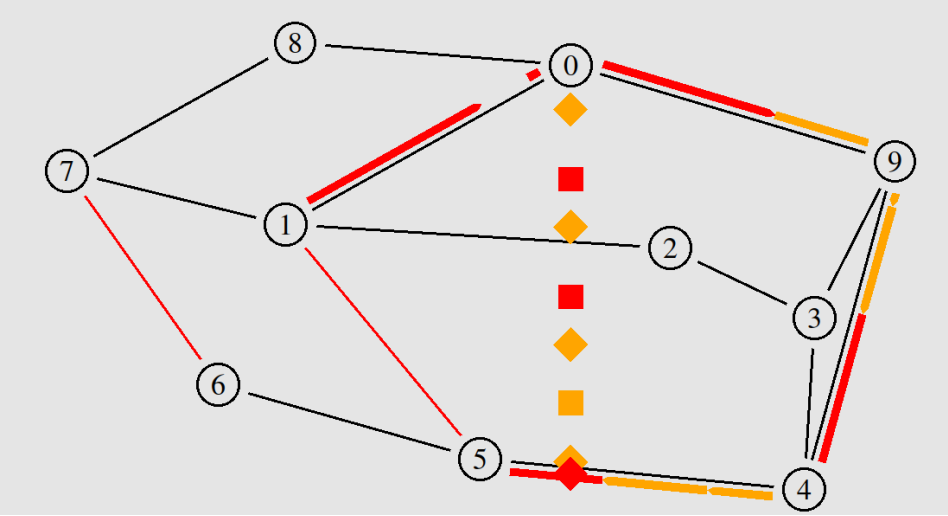
  $ns rtmodel-at 20.0 up $n(7) $n(6)

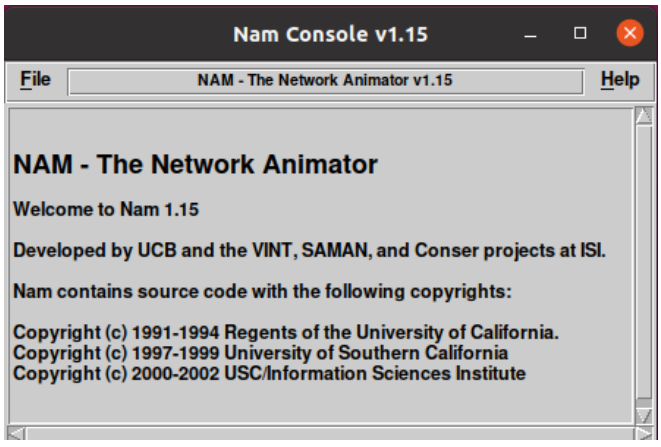
You will see that between the seconds 10.0 and 15.0, the link will be down between node 1 and 5 and node 7 and 6 respectively, so data might get lost (link failure). At 30.0th and 40.0th second, the link will be up from node 1 and 5 again and node 7 and 6 respectively.

SCREENSHOTS:

```
bhavya1022@ubuntu:~$ ns distlink.tcl
 When configured, ns found the right version of tclsh in /usr/bin/tclsh8.6
but it doesn't seem to be there anymore, so ns will fall back on running the fir
st tclsh in your path. The wrong version of tclsh may break the test suites. Rec
onfigure and rebuild ns if this is a problem.
```



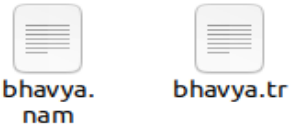


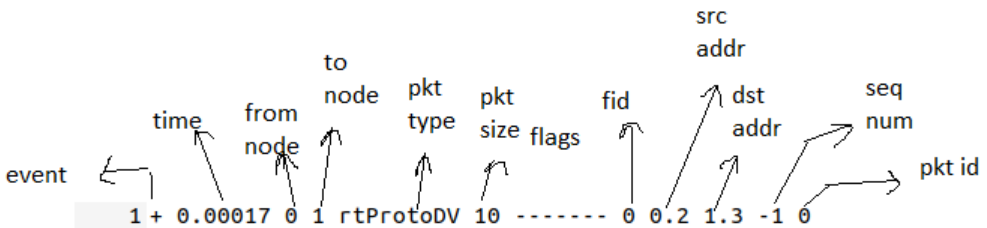


THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Name: Bhavya Sheth

Roll No.: 97






```
  1 + 0.00017 0 1 rtProtoDV 10 ------- 0 0.2 1.3 -1 0
  2 - 0.00017 0 1 rtProtoDV 10 ------- 0 0.2 1.3 -1 0
  3 + 0.00017 0 8 rtProtoDV 10 ------- 0 0.2 8.1 -1 1
  4 - 0.00017 0 8 rtProtoDV 10 ------- 0 0.2 8.1 -1 1
  5 + 0.00017 0 9 rtProtoDV 10 ------- 0 0.2 9.1 -1 2
  6 - 0.00017 0 9 rtProtoDV 10 ------- 0 0.2 9.1 -1 2
  7 + 0.007102 2 1 rtProtoDV 10 ------- 0 2.1 1.3 -1 3
  8 - 0.007102 2 1 rtProtoDV 10 ------- 0 2.1 1.3 -1 3
  9 + 0.007102 2 3 rtProtoDV 10 ------- 0 2.1 3.1 -1 4
 10 - 0.007102 2 3 rtProtoDV 10 ------- 0 2.1 3.1 -1 4
 11 r 0.01025 0 1 rtProtoDV 10 ------- 0 0.2 1.3 -1 0
 12 + 0.01025 1 0 rtProtoDV 10 ------- 0 1.3 0.2 -1 5
 13 - 0.01025 1 0 rtProtoDV 10 ------- 0 1.3 0.2 -1 5
 14 + 0.01025 1 2 rtProtoDV 10 ------- 0 1.3 2.1 -1 6
 15 - 0.01025 1 2 rtProtoDV 10 ------- 0 1.3 2.1 -1 6
 16 + 0.01025 1 5 rtProtoDV 10 ------- 0 1.3 5.4 -1 7
 17 - 0.01025 1 5 rtProtoDV 10 ------- 0 1.3 5.4 -1 7
 18 + 0.01025 1 7 rtProtoDV 10 ------- 0 1.3 7.1 -1 8
 19 - 0.01025 1 7 rtProtoDV 10 ------- 0 1.3 7.1 -1 8
 20 r 0.01025 0 8 rtProtoDV 10 ------- 0 0.2 8.1 -1 1
```

```
1020 r 1.439 0 1 cbr 500 ------- 1 0.0 5.1 85 287
1021 + 1.439 1 5 cbr 500 ------- 1 0.0 5.1 85 287
1022 d 1.439 1 5 cbr 500 ------- 1 0.0 5.1 85 287
1023 r 1.43996 5 1 ack 40 ------- 1 5.0 1.0 27 288
1024 + 1.43996 1 5 tcp 1040 ------- 1 1.0 5.0 47 292
1025 d 1.43996 1 5 tcp 1040 ------- 1 1.0 5.0 47 292
1026 + 1.44 0 1 cbr 500 ------- 1 0.0 5.1 88 293
1027 - 1.44 0 1 cbr 500 ------- 1 0.0 5.1 88 293
1028 - 1.44028 1 5 tcp 1040 ------- 1 1.0 5.0 30 222
1029 r 1.44196 1 5 cbr 500 ------- 1 0.0 5.1 45 213
1030 r 1.444 0 1 cbr 500 ------- 1 0.0 5.1 86 289
1031 + 1.444 1 5 cbr 500 ------- 1 0.0 5.1 86 289
1032 + 1.445 0 1 cbr 500 ------- 1 0.0 5.1 89 294
1033 - 1.445 0 1 cbr 500 ------- 1 0.0 5.1 89 294
1034 r 1.44828 5 1 ack 40 ------- 1 5.0 1.0 28 291
1035 + 1.44828 1 5 tcp 1040 ------- 1 1.0 5.0 48 295
1036 d 1.44828 1 5 tcp 1040 ------- 1 1.0 5.0 48 295
1037 - 1.4486 1 5 cbr 500 ------- 1 0.0 5.1 46 215
1038 r 1.449 0 1 cbr 500 ------- 1 0.0 5.1 87 290
1039 + 1.449 1 5 cbr 500 ------- 1 0.0 5.1 87 290
1040 + 1.45 0 1 cbr 500 ------- 1 0.0 5.1 90 296
```

Conclusion: Thus, Simulation of network with specific routing protocols (Distance vector and link state) was studied and the code was executed. Also, the observation of the traffic flow and the contents of protocol frames was done.