Ex. No. 10	**Routing Protocols**	**Karthik D**
10 October 2021	**UCS1511 - Networks Lab**	**195001047**

Aim

To analyze the difference between the the distance-vector and link-state routing protocols when executing two UDP connections in a network using the Network Simulator (NS2) tool and using a simple tcl-script generated network schedule comprising of FTP traffic using TCP packets

Question

Write a .tcl script to simulate:

1. Create 12 nodes and the links between the nodes as
    a. 0 →8 1Mb 10 ms duplex link droptail
    b. 1 →10 1Mb 10 ms duplex link droptail
    c. 0 →9 1Mb 10 ms duplex link droptail
    d. 9 →11 1Mb 10 ms duplex link droptail
    e. 10 →11 1Mb 10 ms duplex link droptail
    f. 11 →5 1Mb 10 ms duplex link droptail
2. Align all nodes properly
3. Setup a UDP connections over 0 and 5, 1 and 5 with flow id, type, packet size, rate, random fields
4. Set different colors for different flows
5. Use distance vector routing protocol
6. Make links 11-5 and 7-6 down for 1 second
7. Run the simulation for 5 seconds, and show the simulation in network animator and in trace file.•

Similarly write another tcl script to simulate link-state routing protocol

Algorithm

1. Link State Routing

Step 1: Instantiate a new Simulator object. Definite the routing protocol as Link-State (LS). Define colors for different flows for visual distinction

Step 2: Set the simulator to record the simulations in the NAM format. Supply the output file name to be generated at the end

Step 3: Define twelve different nodes in the network simulator — n0 to n11. Create duplex-links as per the specification. All links have the same bandwidth and propagation delays. Use droptail queueing if required

Step 4: Create and attach a UDP agent instance to node n0 and a *null* agent instance to node n5 to receive these packets. Setup a logical connection between the n0 and n5 i.e. set their destination IP and port addresses. Attach a CBR agent as the application layer protocol to the UDP agent.

Step 5: Create and attach a UDP agent instance to node n1 and a *null* agent instance to node n5 to receive these packets. Setup a logical connection between the n1 and n5 i.e. set their destination IP and port addresses. Attach a CBR agent as the application layer protocol to the UDP agent.

Step 6: Set other connection parameters including window size, class, packet size and flow id

Step 7: Schedule the CBR connection to operate from between specific time instances

Step 8: Schedule breakage of links between n5 and n11 as well as n6 and n7 for a 1 second duration

Step 9: Terminate the simulation after 5s. Flush the simulation data collected to the NAM file object created in step-2. Run the generated output trace file using the NAM simulator.

Step 10: Extract the routing packets from the trace file and plot them for later analysis

2. Distance Vector Routing

Step 1: Instantiate a new Simulator object. Definite the routing protocol as Link-State (LS). Define colors for different flows for visual distinction

Step 2: Set the simulator to record the simulations in the NAM format. Supply the output file name to be generated at the end

Step 3: Define twelve different nodes in the network simulator — n0 to n11. Create duplex-links as per the specification. All links have the same bandwidth and propagation delays. Use droptail queueing if required

Step 4: Create and attach a UDP agent instance to node n0 and a *null* agent instance to node n5 to receive these packets. Setup a logical connection between the n0 and n5 i.e. set their destination IP and port addresses. Attach a CBR agent as the application layer protocol to the UDP agent.

Step 5: Create and attach a UDP agent instance to node n1 and a *null* agent instance to node n5 to receive these packets. Setup a logical connection between the n1 and n5 i.e. set their destination IP and port addresses. Attach a CBR agent as the application layer protocol to the UDP agent.

Step 6: Set other connection parameters including window size, class, packet size and flow id

Step 7: Schedule the CBR connection to operate from between specific time instances

Step 8: Schedule breakage of links between n5 and n11 as well as n6 and n7 for a 1 second duration

Step 9: Terminate the simulation after 5s. Flush the simulation data collected to the NAM file object created in step-2. Run the generated output trace file using the NAM simulator.

Step 10: Extract the routing packets from the trace file and plot them for later analysis

TCL Program Code

1. routing-ls.tcl - Trace file and simulation generation script for link-state routing

```
#Create a simulator object
set ns [new Simulator]
$ns rtproto LS

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

# Open the trace file
set nf [open linkstate_out.tr w]
$ns trace-all $nf

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

#Define a 'finish' procedure
proc finish {} {
    global ns nf
```

```
    $ns flush-trace
    #Close the NAM trace file
    close $nf
    # Extract graph points and plot them
    exec awk -f extract_rtpkts.awk linkstate_out.tr > ls_graph.tr &
    exec xgraph ls_graph.tr &
    #Execute NAM on the trace file
    exec nam linkstate_out.nam &
    exit 0
}

#Create twelve 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]
set n10 [$ns node]
set n11 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n8 1Mb 10ms DropTail
$ns duplex-link $n8 $n7 1Mb 10ms DropTail
$ns duplex-link $n7 $n6 1Mb 10ms DropTail
$ns duplex-link $n6 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n10 1Mb 10ms DropTail
$ns duplex-link $n10 $n11 1Mb 10ms DropTail
$ns duplex-link $n11 $n5 1Mb 10ms DropTail
$ns duplex-link $n5 $n4 1Mb 10ms DropTail
$ns duplex-link $n4 $n3 1Mb 10ms DropTail
$ns duplex-link $n3 $n9 1Mb 10ms DropTail
$ns duplex-link $n9 $n0 1Mb 10ms DropTail
$ns duplex-link $n9 $n11 1Mb 10ms DropTail
#Other links to complete the network
```

```
# Set queue-limit on node-11
$ns queue-limit $n10 $n11 50
$ns queue-limit $n5 $n11 50
# Set queue-limit on node-9
$ns queue-limit $n0 $n9 50
$ns queue-limit $n3 $n9 50

#Give node position (for NAM)
#Upper-Ring
$ns duplex-link-op $n0 $n8 orient right-up
$ns duplex-link-op $n8 $n7 orient right
$ns duplex-link-op $n7 $n6 orient right
$ns duplex-link-op $n6 $n2 orient right
$ns duplex-link-op $n2 $n1 orient right-down
$ns duplex-link-op $n1 $n10 orient left-down
$ns duplex-link-op $n10 $n11 orient left
#Lower-Ring
$ns duplex-link-op $n11 $n5 orient down
$ns duplex-link-op $n5 $n4 orient left
$ns duplex-link-op $n4 $n3 orient left-up
$ns duplex-link-op $n3 $n9 orient right-up
$ns duplex-link-op $n9 $n0 orient left-up
$ns duplex-link-op $n9 $n11 orient right

#Monitor the queue for link (n11). (for NAM)
$ns duplex-link-op $n10 $n11 queuePos 0.5
$ns duplex-link-op $n5 $n11 queuePos 0.5
#Monitor the queue for link (n9). (for NAM)
$ns duplex-link-op $n9 $n3 queuePos -0.5
$ns duplex-link-op $n9 $n0 queuePos -0.5

#Setup a UDP connection
set udp_1 [new Agent/UDP]
$udp_1 set class_ 1
#Set the source node
$ns attach-agent $n0 $udp_1
# Set the sink node (null)
set null_1 [new Agent/Null]
$ns attach-agent $n5 $null_1
```

```
# Create a logical connection between the two agents
# Done by assigning respective IPs mutually
$ns connect $udp_1 $null_1
$udp_1 set fid_ 1
#Setup a CBR over UDP connection
set cbr_1 [new Application/Traffic/CBR]
$cbr_1 attach-agent $udp_1
$cbr_1 set type_ CBR
$cbr_1 set packet_size_ 1000
$cbr_1 set rate_ 0.5mb
$cbr_1 set random_ false

#Setup a UDP connection
set udp_2 [new Agent/UDP]
$udp_2 set class_ 2
#Set the source node
$ns attach-agent $n1 $udp_2
# Set the sink node (null)
set null_2 [new Agent/Null]
$ns attach-agent $n5 $null_2
# Create a logical connection between the two agents
# Done by assigning respective IPs mutually
$ns connect $udp_2 $null_2
$udp_2 set fid_ 2
#Setup a CBR over UDP connection
set cbr_2 [new Application/Traffic/CBR]
$cbr_2 attach-agent $udp_2
$cbr_2 set type_ CBR
$cbr_2 set packet_size_ 1000
$cbr_2 set rate_ 0.5mb
$cbr_2 set random_ false

#Schedule events for the UDP agents
$ns at 0.5 "$cbr_1 start"
$ns at 1.0 "$cbr_2 start"
$ns rtmodel-at 2.0 down $n11 $n5
$ns rtmodel-at 2.5 down $n7 $n6
$ns rtmodel-at 3.0 up $n11 $n5
$ns rtmodel-at 3.5 up $n7 $n6
$ns at 4.0 "$cbr_1 stop"
```

```
$ns at 4.5 "$cbr_2 stop"

#Call the finish procedure after 5.5 seconds of simulation time
$ns at 5.5 "finish"

#Run the simulation
$ns run
```

2. routing-dv.tcl - Trace file and simulation generation script for distance-vector routing

```
#Create a simulator object
set ns [new Simulator]
$ns rtproto DV

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

# Open the trace file
set nf [open distvec_out.tr w]
$ns trace-all $nf

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

#Define a 'finish' procedure
proc finish {} {
    global ns nf
    $ns flush-trace
    #Close the NAM trace file
    close $nf
    # Extract graph points and plot them
    exec awk -f extract_rtpkts.awk distvec_out.tr > dv_graph.tr &
    exec xgraph dv_graph.tr &
    #Execute NAM on the trace file
    exec nam distvec_out.nam &
    exit 0
}
```

```
#Create twelve 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]
set n10 [$ns node]
set n11 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n8 1Mb 10ms DropTail
$ns duplex-link $n8 $n7 1Mb 10ms DropTail
$ns duplex-link $n7 $n6 1Mb 10ms DropTail
$ns duplex-link $n6 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n10 1Mb 10ms DropTail
$ns duplex-link $n10 $n11 1Mb 10ms DropTail
$ns duplex-link $n11 $n5 1Mb 10ms DropTail
$ns duplex-link $n5 $n4 1Mb 10ms DropTail
$ns duplex-link $n4 $n3 1Mb 10ms DropTail
$ns duplex-link $n3 $n9 1Mb 10ms DropTail
$ns duplex-link $n9 $n0 1Mb 10ms DropTail
$ns duplex-link $n9 $n11 1Mb 10ms DropTail
#Other links to complete the network

# Set queue-limit on node-11
$ns queue-limit $n10 $n11 50
$ns queue-limit $n5 $n11 50
# Set queue-limit on node-9
$ns queue-limit $n0 $n9 50
$ns queue-limit $n3 $n9 50

#Give node position (for NAM)
#Upper-Ring
```

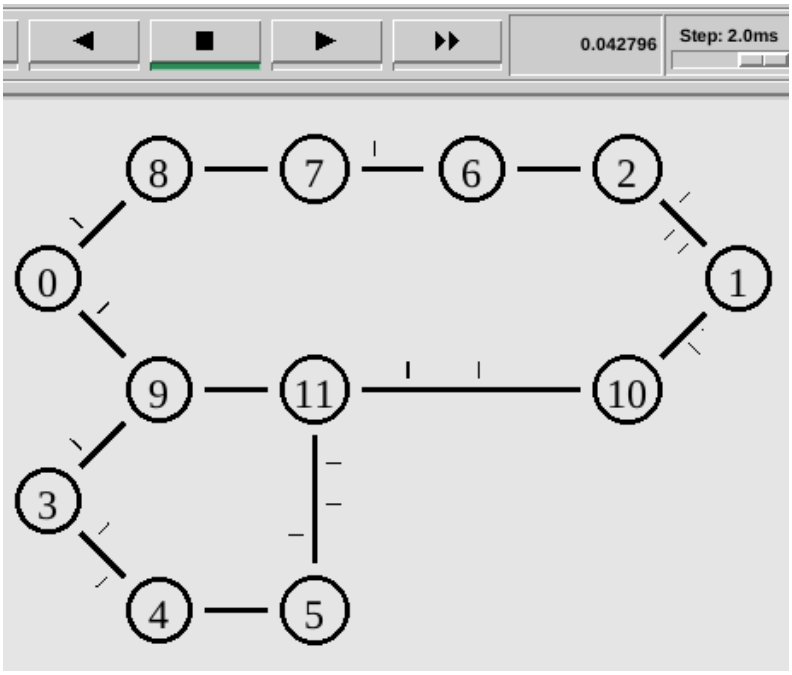
```
$ns duplex-link-op $n0 $n8 orient right-up
$ns duplex-link-op $n8 $n7 orient right
$ns duplex-link-op $n7 $n6 orient right
$ns duplex-link-op $n6 $n2 orient right
$ns duplex-link-op $n2 $n1 orient right-down
$ns duplex-link-op $n1 $n10 orient left-down
$ns duplex-link-op $n10 $n11 orient left
#Lower-Ring
$ns duplex-link-op $n11 $n5 orient down
$ns duplex-link-op $n5 $n4 orient left
$ns duplex-link-op $n4 $n3 orient left-up
$ns duplex-link-op $n3 $n9 orient right-up
$ns duplex-link-op $n9 $n0 orient left-up
$ns duplex-link-op $n9 $n11 orient right

#Monitor the queue for link (n11). (for NAM)
$ns duplex-link-op $n10 $n11 queuePos 0.5
$ns duplex-link-op $n5 $n11 queuePos 0.5
#Monitor the queue for link (n9). (for NAM)
$ns duplex-link-op $n9 $n3 queuePos -0.5
$ns duplex-link-op $n9 $n0 queuePos -0.5

#Setup a UDP connection
set udp_1 [new Agent/UDP]
$udp_1 set class_ 1
#Set the source node
$ns attach-agent $n0 $udp_1
# Set the sink node (null)
set null_1 [new Agent/Null]
$ns attach-agent $n5 $null_1
# Create a logical connection between the two agents
# Done by assigning respective IPs mutually
$ns connect $udp_1 $null_1
$udp_1 set fid_ 1
#Setup a CBR over UDP connection
set cbr_1 [new Application/Traffic/CBR]
$cbr_1 attach-agent $udp_1
$cbr_1 set type_ CBR
$cbr_1 set packet_size_ 1000
$cbr_1 set rate_ 0.5mb
```

```
$cbr_1 set random_ false

#Setup a UDP connection
set udp_2 [new Agent/UDP]
$udp_2 set class_ 2
#Set the source node
$ns attach-agent $n1 $udp_2
# Set the sink node (null)
set null_2 [new Agent/Null]
$ns attach-agent $n5 $null_2
# Create a logical connection between the two agents
# Done by assigning respective IPs mutually
$ns connect $udp_2 $null_2
$udp_2 set fid_ 2
#Setup a CBR over UDP connection
set cbr_2 [new Application/Traffic/CBR]
$cbr_2 attach-agent $udp_2
$cbr_2 set type_ CBR
$cbr_2 set packet_size_ 1000
$cbr_2 set rate_ 0.5mb
$cbr_2 set random_ false

#Schedule events for the UDP agents
$ns at 0.5 "$cbr_1 start"
$ns at 1.0 "$cbr_2 start"
$ns rtmodel-at 2.0 down $n11 $n5
$ns rtmodel-at 2.5 down $n7 $n6
$ns rtmodel-at 3.0 up $n11 $n5
$ns rtmodel-at 3.5 up $n7 $n6
$ns at 4.0 "$cbr_1 stop"
$ns at 4.5 "$cbr_2 stop"

#Call the finish procedure after 5.5 seconds of simulation time
$ns at 5.5 "finish"

#Run the simulation
$ns run
```

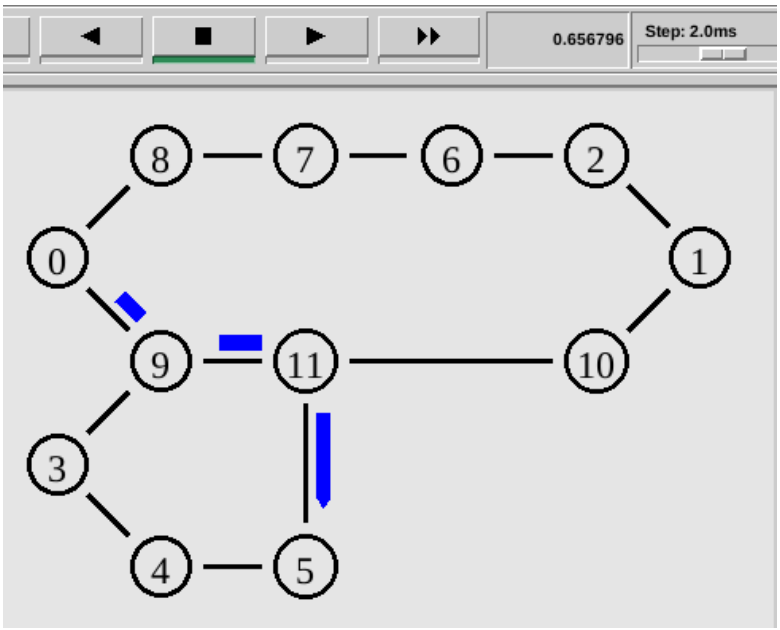
Sample Output

1. **Link State Routing**

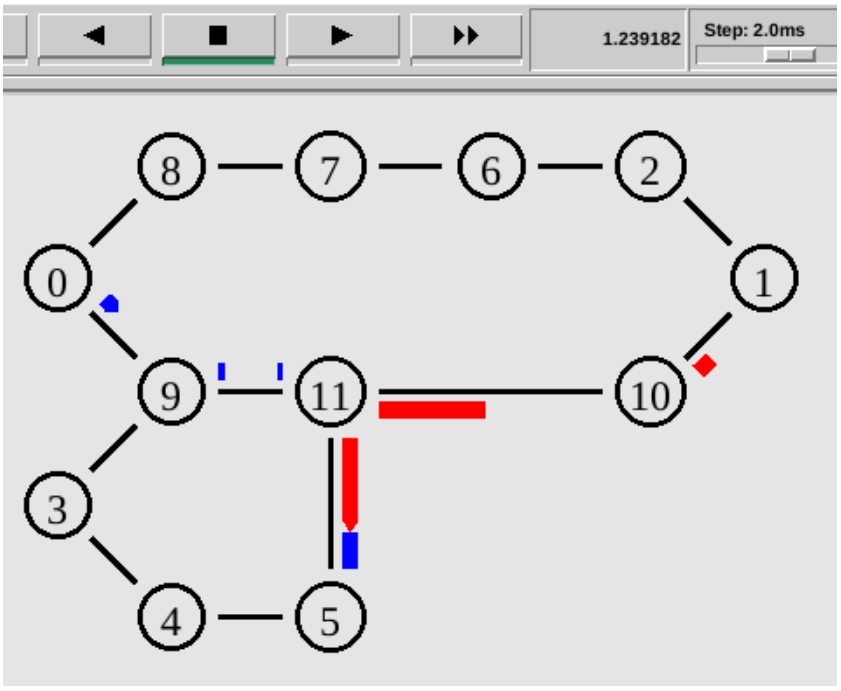
   a. **Initial State - Nodes Exchange State Information (Flooding)**



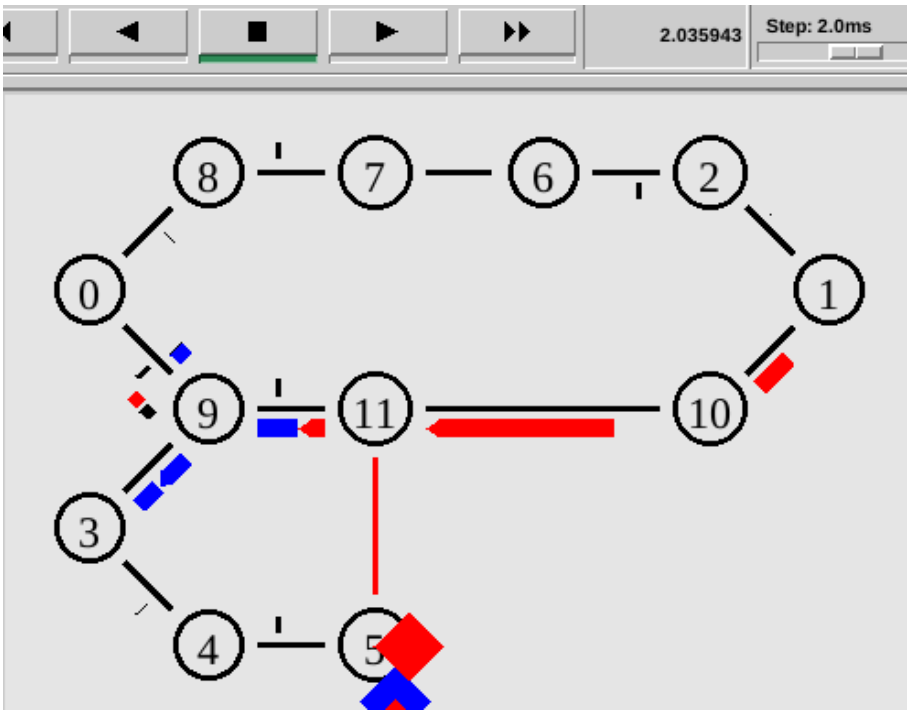
   b. **Transmission from node-0 to node-5 starts**



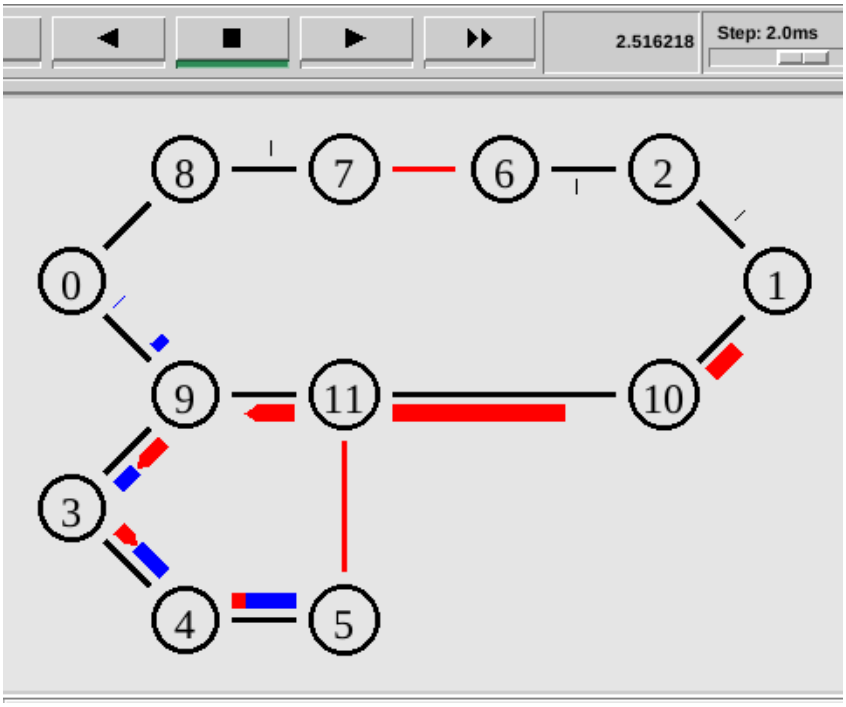
c. **Transmission from node-1 to node-5 starts**



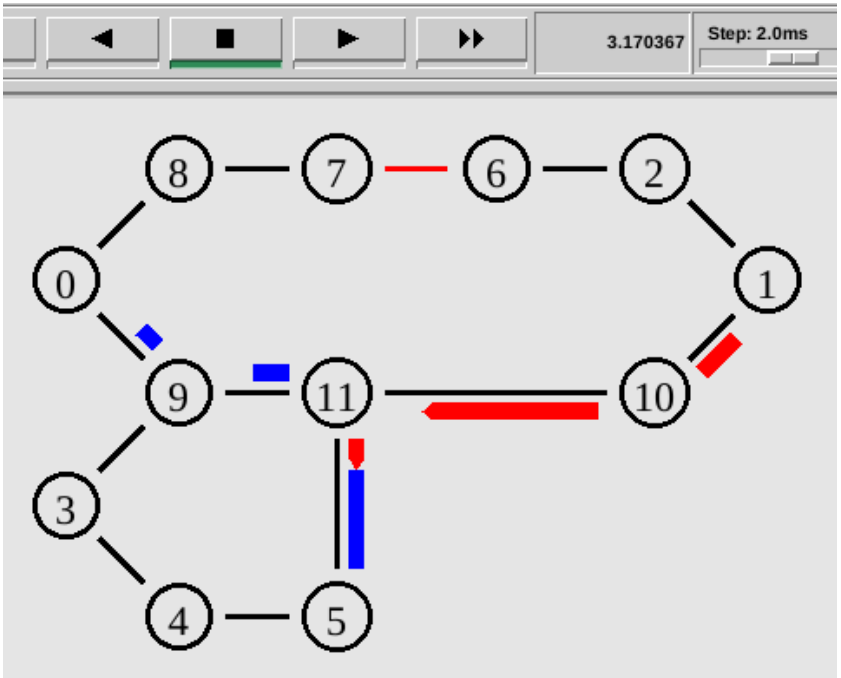
d. **Soon after link b/w nodes 11 and 5 is disabled, flooding occurs to propagate this information (black colored packets). Packets queued at node-11 dropped**



e. **Both connections are rerouted through 11-9-3-4-5. Flooding again occurs when link between nodes 6 and 7 is disabled**

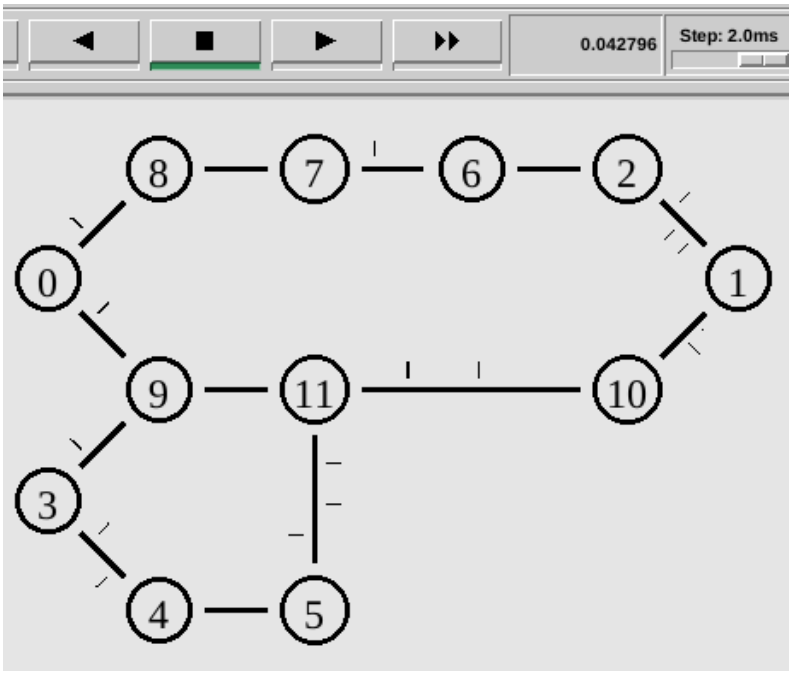


f. **Original route is restored in both connections when the link between is nodes 11 and 5 is back up**

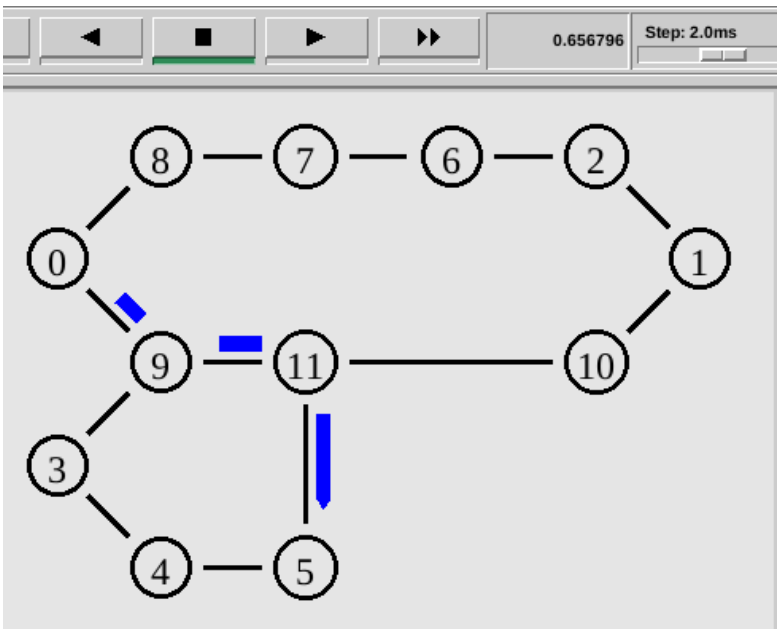


2. **Distance Vector Routing**

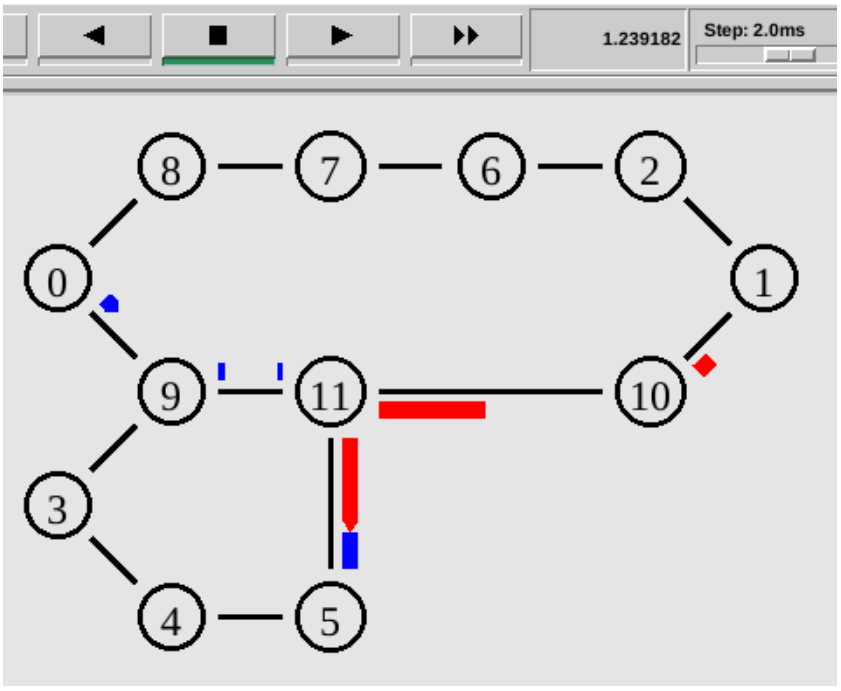
   a. **Initial State - Nodes Exchange State Information as distance vector - Smaller packets**



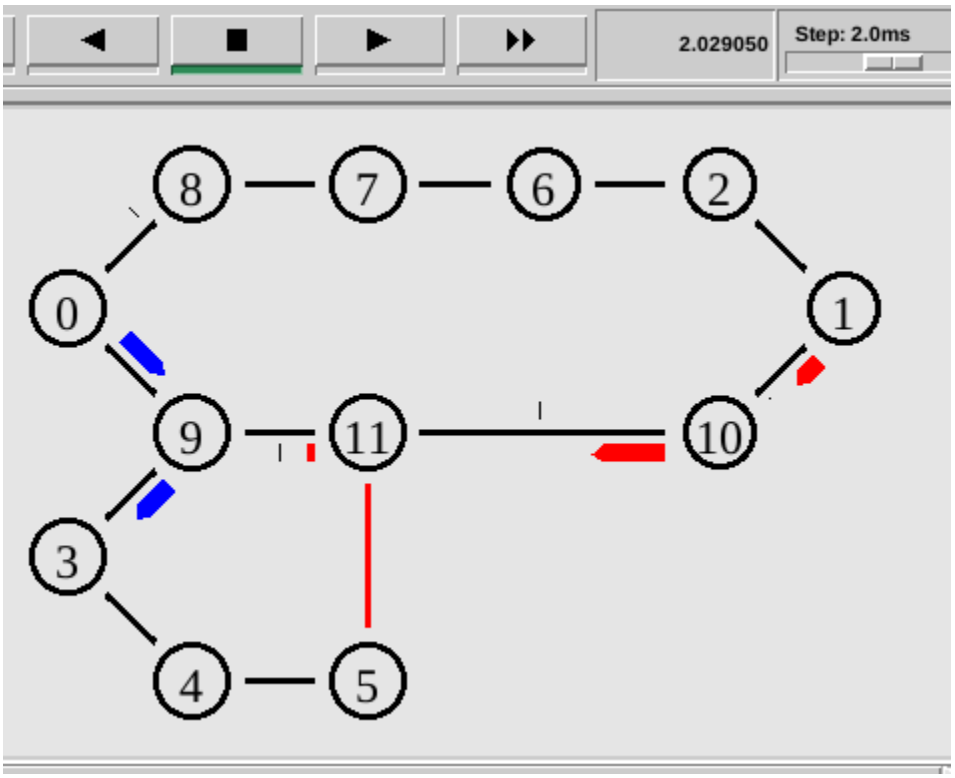
   b. **Transmission from node-0 to node-5 starts**



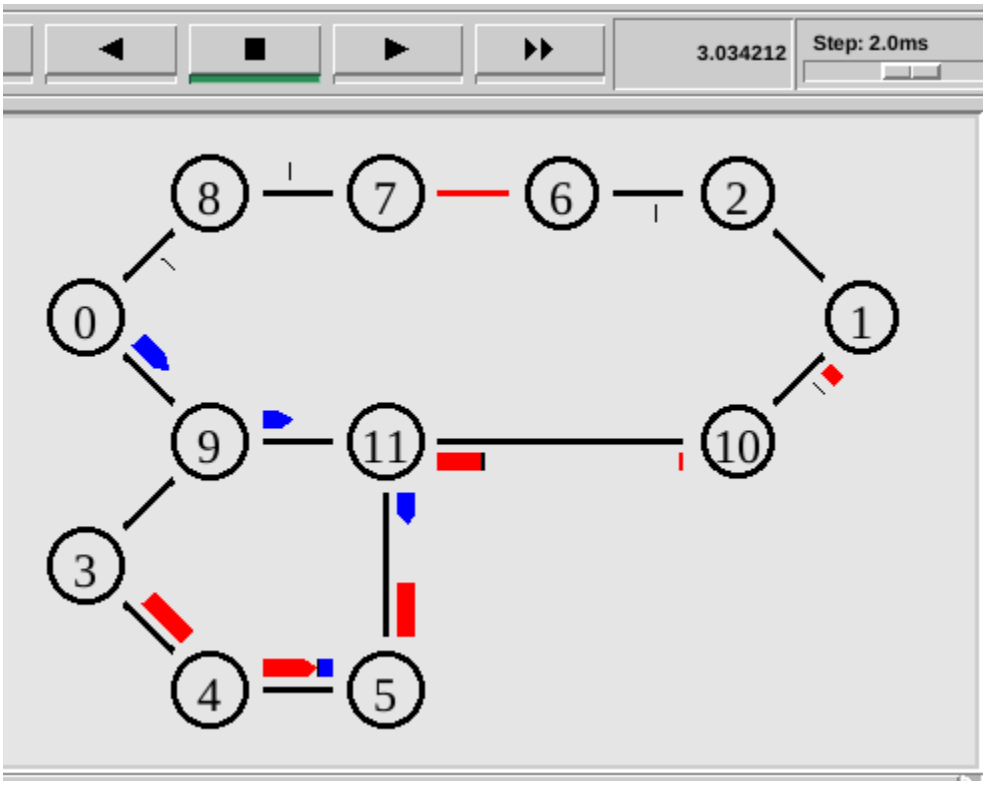
c. **Transmission from node-1 to node-5 starts**



d. **Soon after link b/w nodes 11 and 5 is disabled, distance vectors are shared between neighbors to propagate this information (black colored packets)**



e. **Original route is restored in both connections when the link between nodes 11 and 5 is back up. Again, neighbors share distance vectors (black packets)**

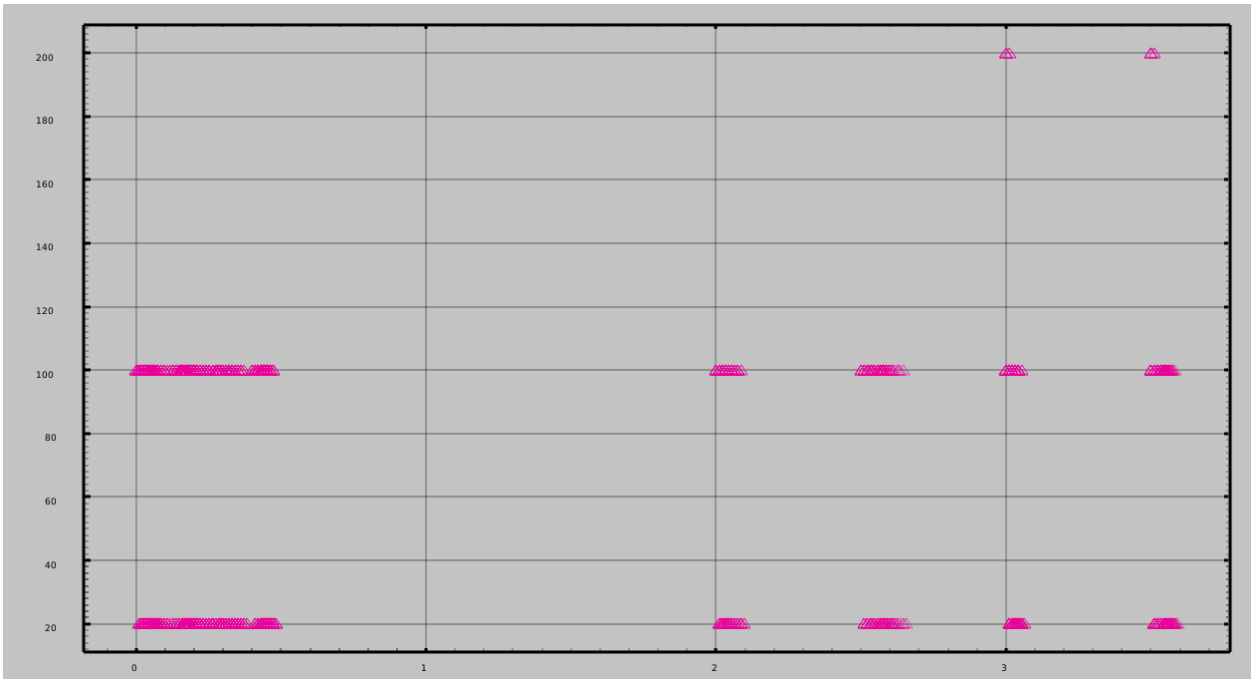


<u>Differences Observed between Link-State (LS) and Distance-Vector (DV)</u>

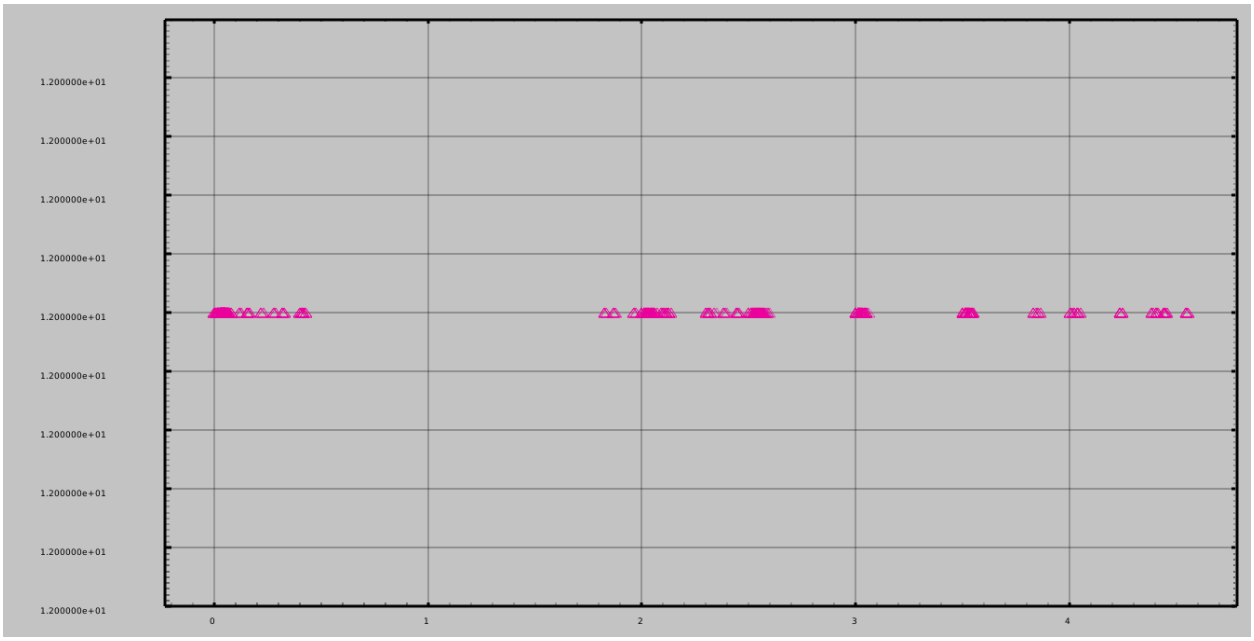
1. LS transmits larger packets detailing the links' state and each node propagates a received packet to all nodes except the source node. DV uses only a single distance vector and is sent only to its neighbors. Hence, smaller packet size and no flooding is observed

2. Note the graphs below:

a. LS Graph (Routing Packet Size Vs. Time of Packet Transmission)



b. DV Graph (Routing Packet Size Vs. Time of Packet Transmission)



LS is characterized by large number of packets of higher size (20 bytes and 100 bytes) being transmitted, each time a new network is constructed and when links are enabled and disabled (eg. at time=2s, link 11-5 was disabled). However, it converges and understands the

network much quicker (shorter length of routing packet transmission phase - measured along the time axis)

On the contrary, DV has a much smaller packet size (only 12 bytes!) and a lesser number of interchanges. But it takes much longer to converge and understand the changes in the network (longer length of routing packet transmission phase - measured along the time axis)

Result

Analyzed the difference between the link-state and distance-vector routing protocols using two UDP agents using the Network Simulator (NS2) tool and using a simple tcl-script generated network schedule consisting of CBR traffic. Through this implementation, the following aspects were understood:

1. Basic scripting for NS and NAM using TCL language
2. Difference between the link-state and distance-vector routing protocols
3. Working with NS2 and NAM to observe network traffic conceptually
4. Working with Xgraph and awk to extract and plot trace information