National University of Computer and Emerging Sciences



**Laboratory Manuals**

*for*

**Computer Networks - Lab**

(CL -3001)

|  |  |
| --- | --- |
| Files to be submitted: | 1.Word File with tcl code+Screenshots  2. TCL code files |
| File Names: | RollNo\_Sec1/2\_NameOnFlex |

*Department of Computer Science*

*FAST-NU, Lahore, Pakistan*

Lab Manual 10

# Objective:

* Analyze Distance vector routing protocol on NS2
* Implementation of Circular and Star Topology using ns2

**Distance vector routing protocol**

**Distance vector protocols** (a **vector** contains both **distance** and direction) determines the path to remote networks using hop count as the metric. A hop count is defined as the number of times a packet needs to pass through a **router** to reach a remote (far away) destination.

In NS2, we add following command to set the routing protocol to Distance Vector

***$ns rtproto* DV**

To bring the link between two nodes down/up at specific simulation time we write the following commands  
**$ns rtmodel-at 0.30 down $node1 $*node2***

***$ns rtmodel-at 0.40 up $node1 $node2***

**For Arrays in tcl:**

**set arrayName(Index) arrayValue**

**NOTE:**

* You will have to implement both the following topologies using For loop in TCL Language and use as much less statements as possible.
* You must define the nodes and links between the nodes using for loop. Simple definitions of nodes/links will not be accepted.

**In-lab statement 1: [10]**

1. Write a Tcl script that forms a network consisting of **7 nodes**, numbered from 0 to 6, forming a **ring topology.**
2. The links have a 512Kbps bandwidth with 5ms delay and droptail queue.
3. Set the routing protocol to DV (Distance vector).
4. Send UDP packets from node 0 to node 3 with the rate of 100 packets/sec with each packet having a size of 1 Kilo Bytes.
5. Start transmission at 0.02.
6. Bring **down the link** between node 2 and node 3 at 0.4.
7. Bring the dropped link back **up** at 1.0.
8. Finish the transmission at 1.5

End the simulation at 2.0.

1. Also monitor two queues (b/w Node 0 &1) and (b/w Node 0 &6) and answer the following.

**Answer the following in a your document:**

1. What path do the packets follow initially? And why?
2. What path do the packets take after the link fails? And why?
3. If we remove the Distance Vector Algorithm, what path the packets follow after the link fails? And why?

**Part a**. The packets follow the ring topology from Node 0 to Node 3. The reason is that traversing links in sequence (0-1-2-3).

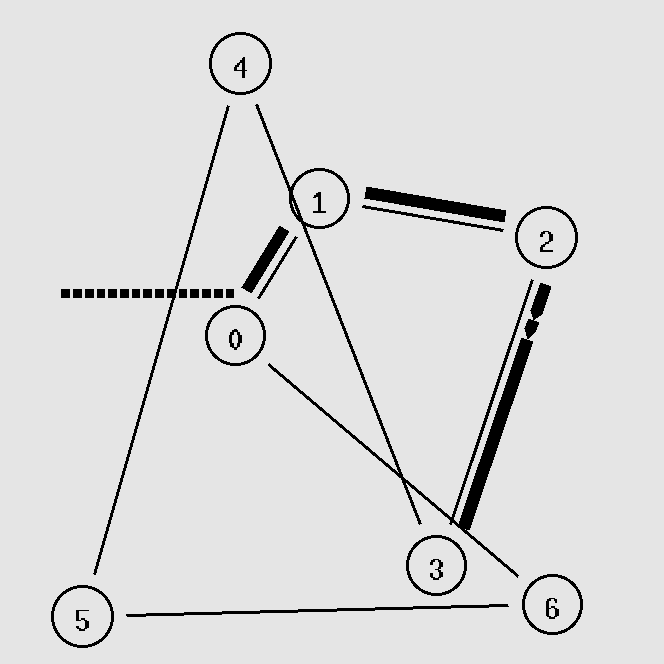
**Part b**. After the link between Node 2 and Node 3 fails at 0.4, the packets take other path through the remaining links in the network (0-6-5-4-3). The reason is that the Distance Vector algorithm recalculates the shortest paths based on the updated topology, and it chooses the available path with the lowest cost.

**Part c**. If the Distance Vector algorithm is removed, the packets might continue to follow the initially calculated path even after the link failure. Without the dynamic adjustment provided by the Distance Vector algorithm, the routing decisions remain static, and the packets could still be sent along the original path (0-1-2-3) despite the link failure.

**Code:**

set ns [new Simulator]   
set tracefile [open [out.tr](http://out.tr/) w]  
$ns trace-all $tracefile  
  
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]  
  
$ns duplex-link $n0 $n1 512Kbps 5ms DropTail  
$ns duplex-link $n1 $n2 512Kbps 5ms DropTail  
$ns duplex-link $n2 $n3 512Kbps 5ms DropTail  
$ns duplex-link $n3 $n4 512Kbps 5ms DropTail  
$ns duplex-link $n4 $n5 512Kbps 5ms DropTail  
$ns duplex-link $n5 $n6 512Kbps 5ms DropTail  
$ns duplex-link $n6 $n0 512Kbps 5ms DropTail  
  
$ns rtproto DV $n0 $n1  
$ns rtproto DV $n1 $n2  
$ns rtproto DV $n2 $n3  
$ns rtproto DV $n3 $n4  
$ns rtproto DV $n4 $n5  
$ns rtproto DV $n5 $n6  
$ns rtproto DV $n6 $n0  
  
set udp [new Agent/UDP]  
$ns attach-agent $n0 $udp  
  
set cbr [new Application/Traffic/CBR]  
$cbr set packetSize 1000  
$cbr set rate 100  
$cbr attach-agent $udp  
$ns at 0.02 "$cbr start"  
  
$ns at 0.4 "$ns duplex-link-op $n2 $n3 orient down"  
  
$ns at 1.0 "$ns duplex-link-op $n2 $n3 orient up"  
$ns at 1.5 "$cbr stop"  
$ns at 2.0 "finish"  
$ns at 2.0 "exit"   
$ns run

**Screenshot**



**In-lab statement 2: [10]**

You will have to create **a star topolgy** as given in the diagram below using ns2 to implement the Distance vector routing protocol. Assume all the devices in the following star topology as nodes and all the wires as duplex links having a **capacity of 512Kb** and a **propagation delay of 10ms** with a **stochastic fair queue** scheduling algorithm.

You will have to send TCP data from H1 to H4 having red color. Also you will have to send UDP data with a rate of 256Kbps from H2 to H5 having blue color.

**Scheduling Events:**

* TCP Data starts at 0.1 and stops at 1.5
* UDP Data starts at 0.2 and stops at 1.3
* Bring the link between SW1 and H5 down at 0.5 and bring it back up at 0.9
* Bring the link between SW1 and H4 down at 0.7 and bring it back up at 1.2
* Stop the simulation at 2.0

**CODE:**

set ns [new Simulator]

set h1 [$ns node]

set h2 [$ns node]

set h4 [$ns node]

set h5 [$ns node]

set sw1 [$ns node]

$ns duplex-link $h1 $sw1 512Kb 10ms DropTail

$ns duplex-link $h2 $sw1 512Kb 10ms DropTail

$ns duplex-link $h4 $sw1 512Kb 10ms DropTail

$ns duplex-link $h5 $sw1 512Kb 10ms DropTail

$ns duplex-link-op $h1 $sw1 orient right-down

$ns duplex-link-op $h2 $sw1 orient left-down

$ns duplex-link-op $h4 $sw1 orient right-up

$ns duplex-link-op $h5 $sw1 orient left-up

$ns queue-limit $h1 $sw1 10

$ns queue-type $h1 $sw1 RED

$ns RED $h1 $sw1

$ns at 0.1 "$h1 attach-tcp $h4 512Kb 10ms"

$ns at 0.2 "$h2 attach-agent udp"

$ns at 0.2 "$h2 set cbr-traffic-source [new Application/Traffic/CBR]"

$ns at 0.2 "$h2 cbr-traffic-source set rate\_ 256Kb"

$ns at 1.5 "$h1 detach-tcp $h4"

$ns at 0.5 "$ns duplex-link-down $sw1 $h5"

$ns at 0.9 "$ns duplex-link-up $sw1 $h5"

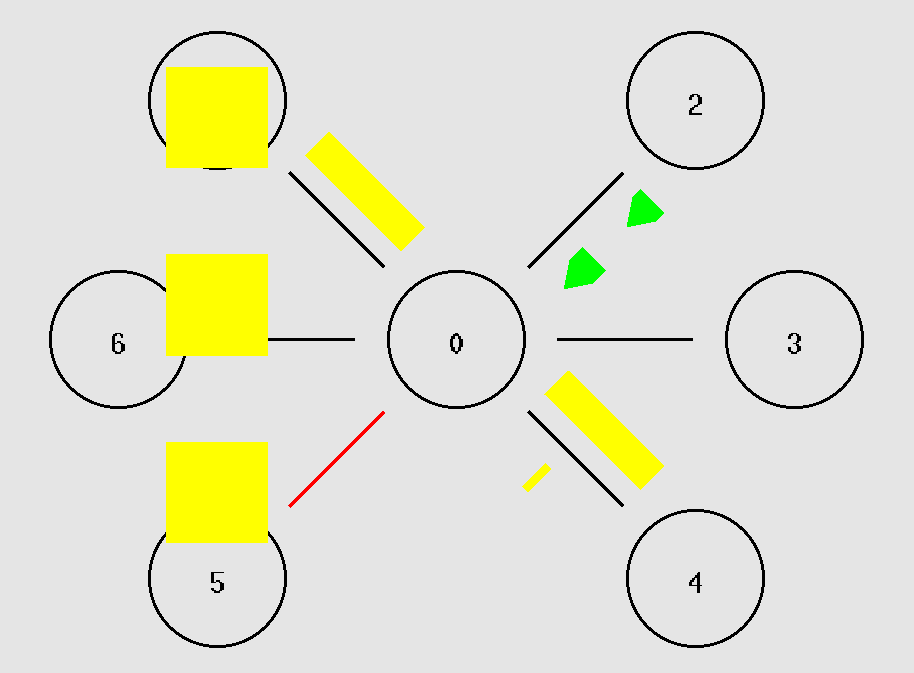
$ns at 0.7 "$ns duplex-link-down $sw1 $h4"

$ns at 1.2 "$ns duplex-link-up $sw1 $h4"

$ns at 2.0 "$ns halt"

$ns run

**SCREENSHOT**



**Note: You must orient the nodes as shown in the topology using the orient feature for NAM. Consider SW1 as a node.**