SSN COLLEGE OF ENGINEERING, KALAVAKKAM

(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1511 – COMPUTER NETWORKS LAB

Lab Exercise 10: Simulation of routing protocol

Aim:

Write tcl script to simulate the routing protocols in wired networks

Algorithm:

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. Also show the flooding in the simulation.

Explanation:

Distance vector routing:

- It is a dynamic routing algorithm in which each router computes distance between itself and each possible destination i.e. its immediate neighbors.
- The router share its knowledge about the whole network to its neighbors and accordingly updates table based on its neighbors.
- The sharing of information with the neighbors takes place at regular intervals.
- It makes use of **Bellman Ford Algorithm** for making routing tables.

- **Problems –** Count to infinity problem which can be solved by splitting horizon.
  – Good news spread fast and bad news spread slowly.
  – Persistent looping problem i.e. loop will be there forever.

Link state routing:

- It is a dynamic routing algorithm in which each router shares knowledge of its neighbors with every other router in the network.
- A router sends its information about its neighbors only to all the routers through flooding.
- Information sharing takes place only whenever there is a change.
- It makes use of **Dijkastra's Algorithm** for making routing tables.
- **Problems –** Heavy traffic due to flooding of packets.
  – Flooding can result in infinite looping which can be solved by using **Time to live (TTL)** field.

Code:

dv.tcl

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

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

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

#Define a 'finish' procedure
proc finish { } {
global ns nr nf
$ns flush-trace
#Close the NAM file
close $nf
#Close the trace file
close $nr
#Execute NAM on the trace file
exec nam dv.nam &
exit 0
}

#Create twelve nodes
for { set i 0 } { $i < 12} { incr i 1 } { set n($i) [$ns node]}

#Create links between the nodes
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(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

#Give node position (for NAM)
$ns duplex-link-op $n(0) $n(1) orient right
$ns duplex-link-op $n(1) $n(2) orient down
$ns duplex-link-op $n(2) $n(3) orient down
$ns duplex-link-op $n(3) $n(4) orient left
$ns duplex-link-op $n(4) $n(5) orient left
$ns duplex-link-op $n(5) $n(6) orient left
$ns duplex-link-op $n(6) $n(7) orient left
$ns duplex-link-op $n(7) $n(8) orient up
$ns duplex-link-op $n(8) $n(0) orient up
$ns duplex-link-op $n(5) $n(11) orient up

$ns duplex-link-op $n(0) $n(9) orient right-down
$ns duplex-link-op $n(9) $n(11) orient right
$ns duplex-link-op $n(11) $n(10) orient right
$ns duplex-link-op $n(1) $n(10) orient left-down

#Setup a UDP connection between node 0 and node 5
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

#Setup a UDP connection between node 1 and node 5
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 null1 [new Agent/Null]
$ns attach-agent $n(5) $null1
$ns connect $udp1 $null1

#Distance Vector Routing
$ns rtproto DV

#Make link 11-5 and 7-6 down for 1 second
$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
$ns at 1.0 "$cbr0 start"
$ns at 2.0 "$cbr1 start"
$ns at 45 "finish"

#Run the simulation
$ns run
```

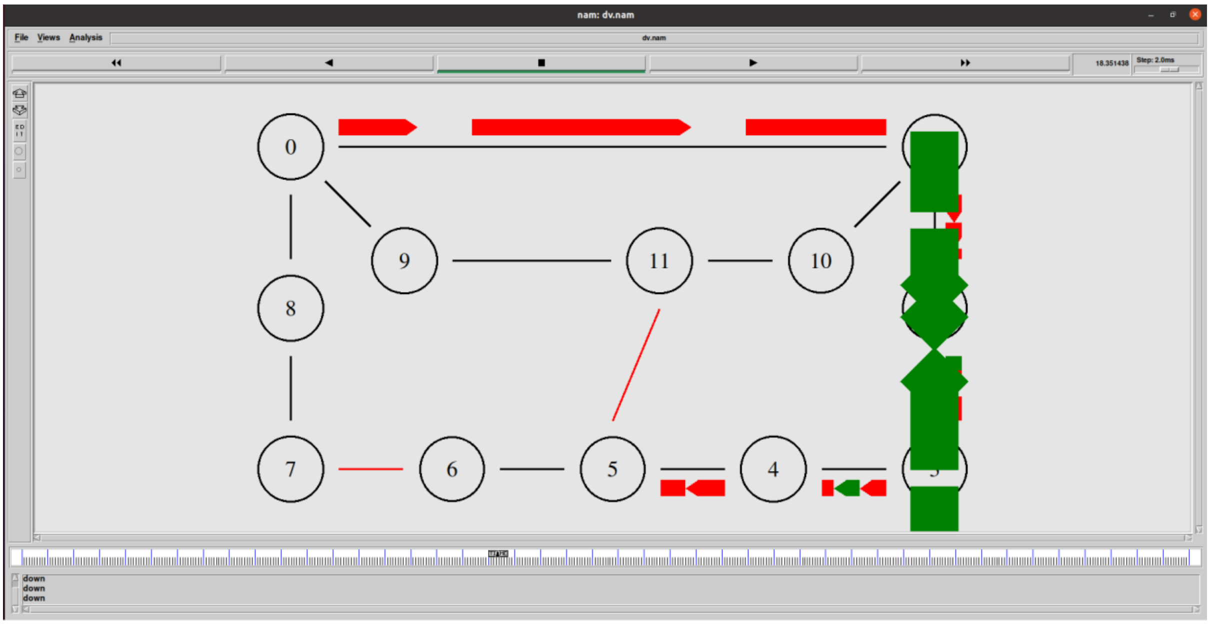
ls.tcl

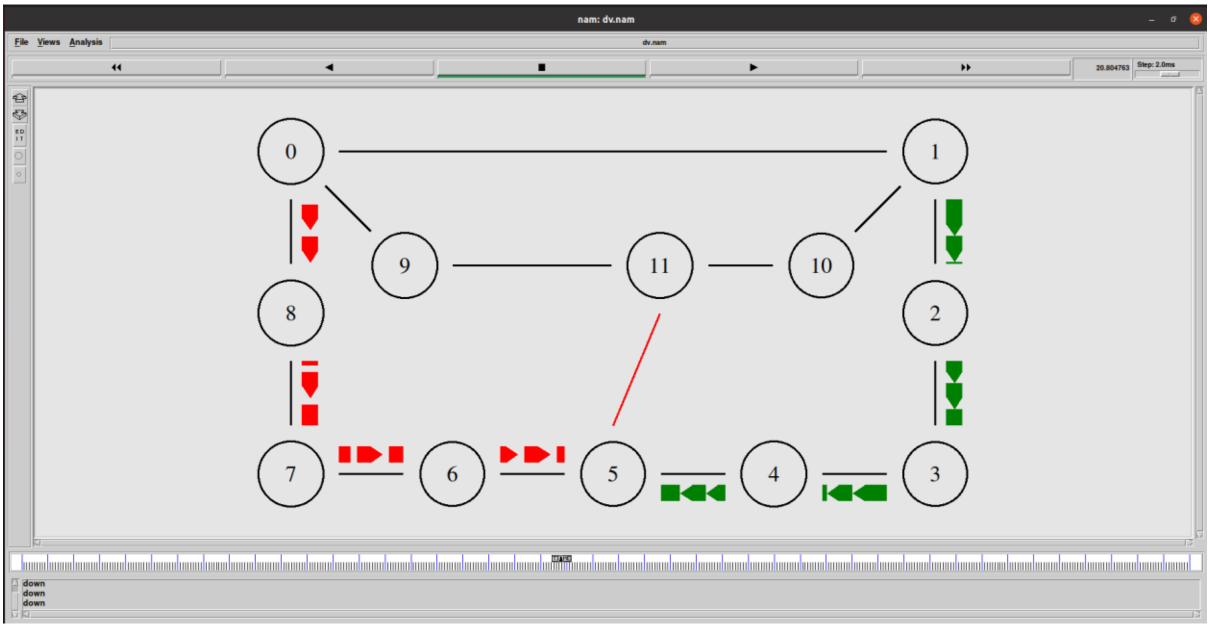
```
#Create a simulator object
set ns [new Simulator]

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

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

#Define a 'finish' procedure
proc finish { } {
global ns nr nf
$ns flush-trace
#Close the NAM file
close $nf
#Close the trace file
close $nr
#Execute NAM on the trace file
exec nam ls.nam &
exit 0
}

#Create twelve nodes
for { set i 0 } { $i < 12} { incr i 1 } { set n($i) [$ns node]}

#Create links between the nodes
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(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

#Give node position (for NAM)
$ns duplex-link-op $n(0) $n(1) orient right
$ns duplex-link-op $n(1) $n(2) orient down
$ns duplex-link-op $n(2) $n(3) orient down
$ns duplex-link-op $n(3) $n(4) orient left
$ns duplex-link-op $n(4) $n(5) orient left
$ns duplex-link-op $n(5) $n(6) orient left
$ns duplex-link-op $n(6) $n(7) orient left
$ns duplex-link-op $n(7) $n(8) orient up
```

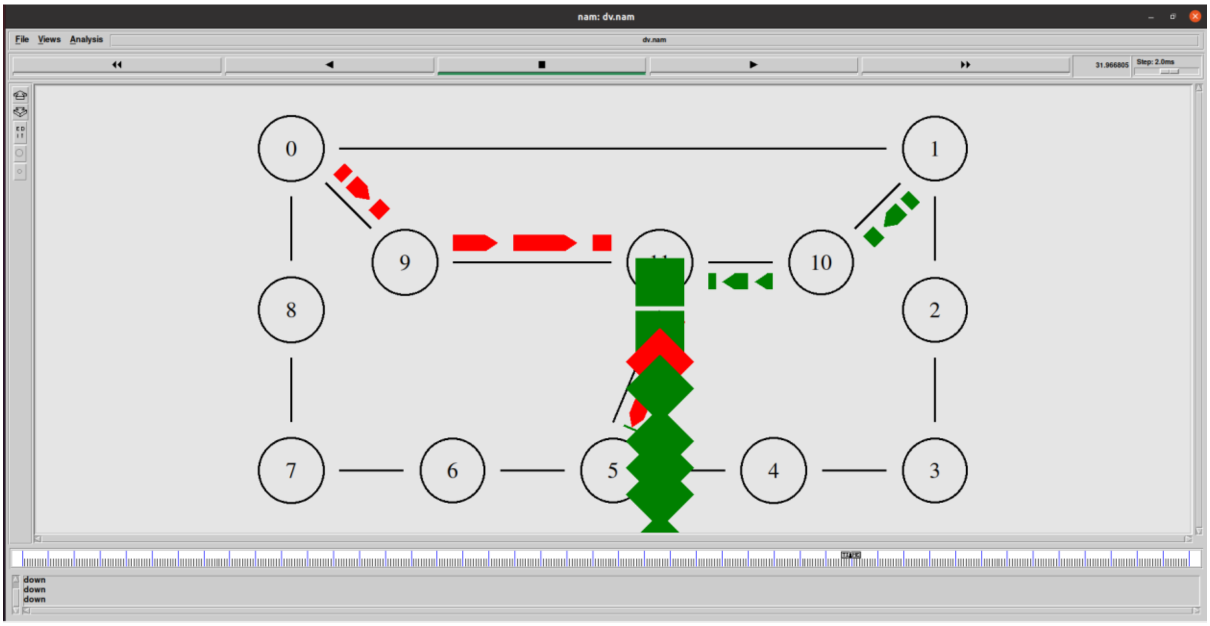
```
$ns duplex-link-op $n(8) $n(0) orient up
$ns duplex-link-op $n(5) $n(11) orient up

$ns duplex-link-op $n(0) $n(9) orient right-down
$ns duplex-link-op $n(9) $n(11) orient right
$ns duplex-link-op $n(11) $n(10) orient right
$ns duplex-link-op $n(1) $n(10) orient left-down

#Setup a UDP connection between node 0 and node 5
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

#Setup a UDP connection between node 1 and node 5
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 null1 [new Agent/Null]
$ns attach-agent $n(5) $null1
$ns connect $udp1 $null1

#Link State Routing
$ns rtproto LS

#Make link 11-5 and 7-6 down for 1 second
$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
$ns at 1.0 "$cbr0 start"
$ns at 2.0 "$cbr1 start"
$ns at 45 "finish"

#Run the simulation
$ns run
```

Output:

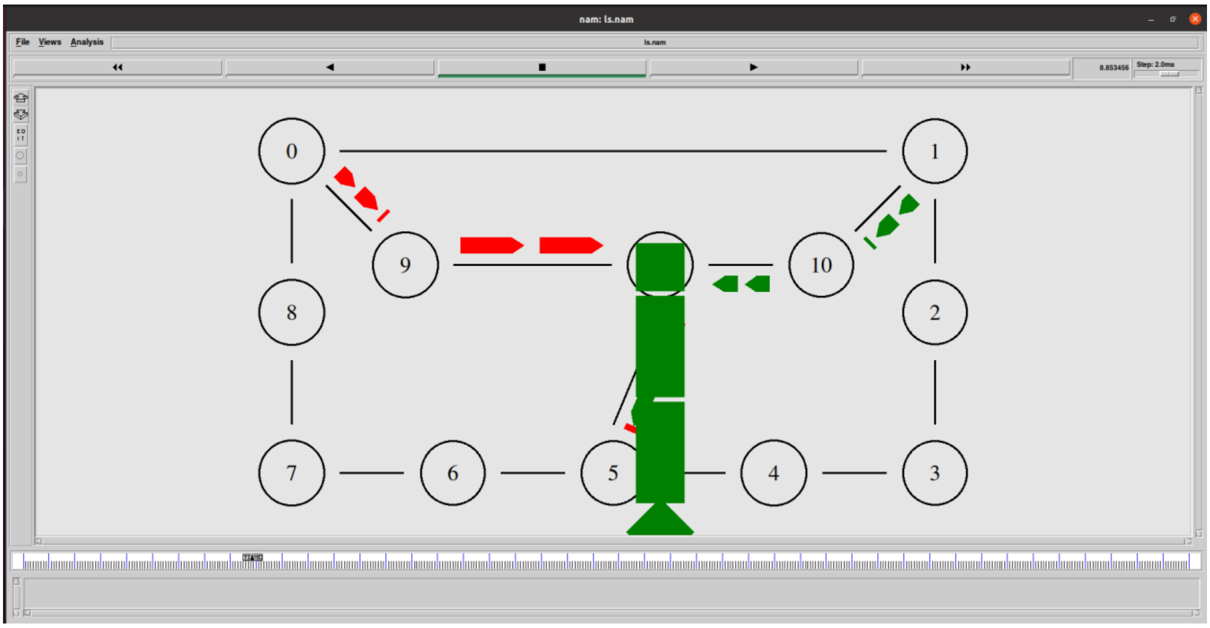
dv.tcl

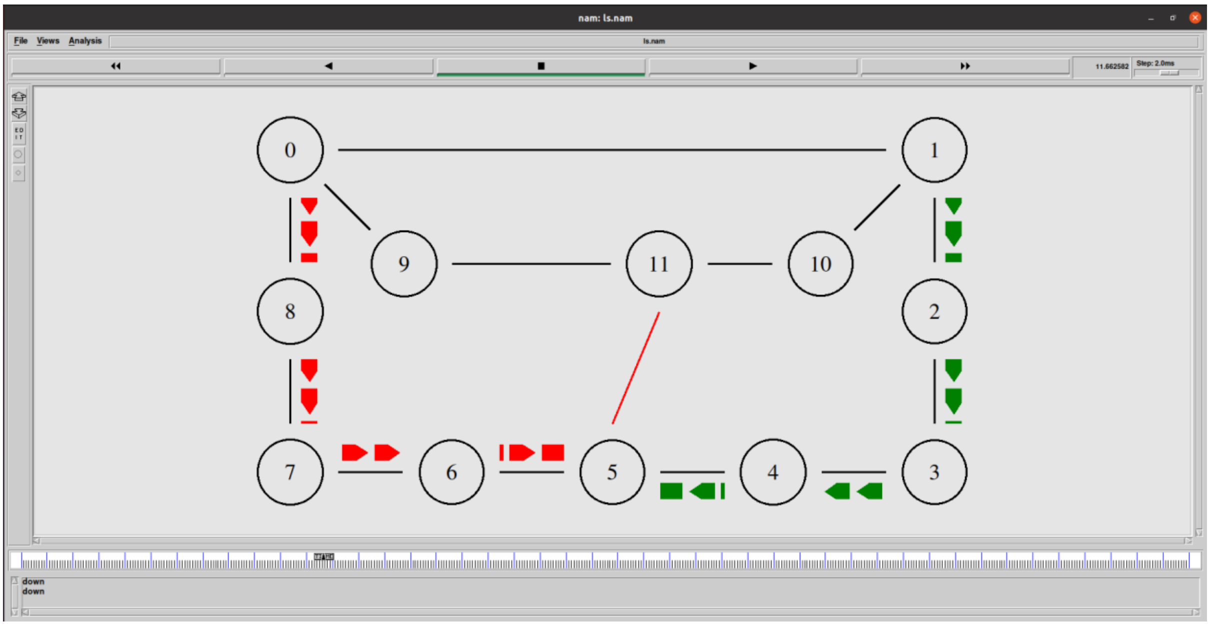


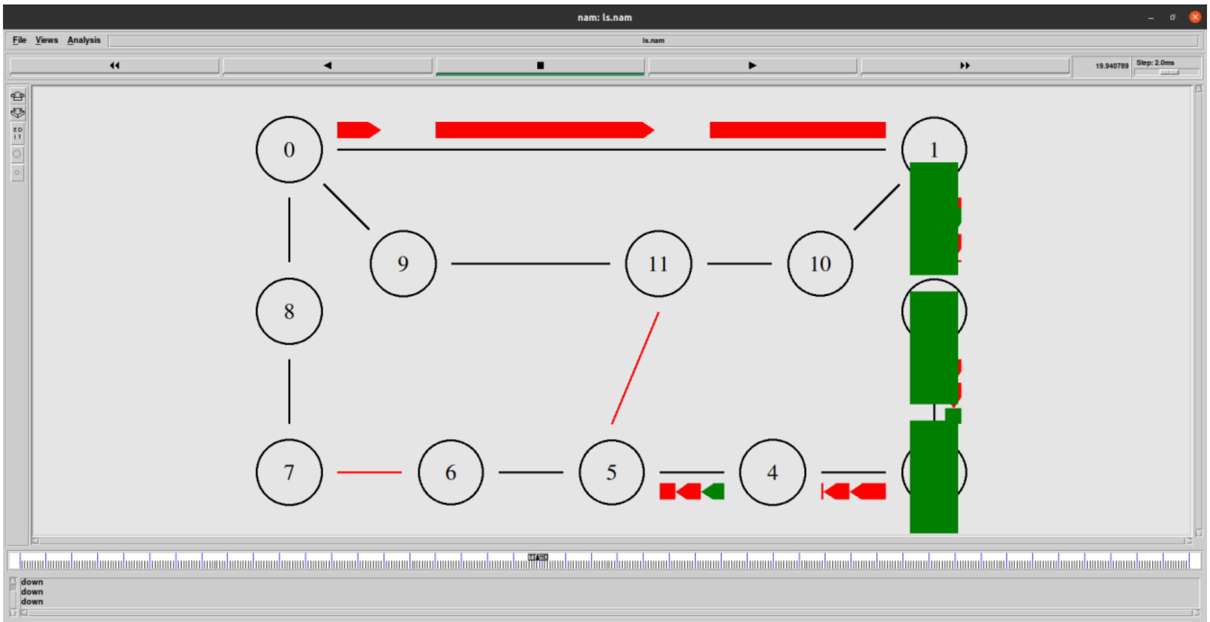



nam: dv.nam
File
Views
Analysis
dv.nam
0
1
9
10
8
2
7
6
5
4
3
down
down
down

ls.tcl


nam: ls.nam
File
Views
Analysis
ls.nam
0
1
9
10
8
2
7
6
5
4
3





Learning outcomes:

1. Reexplored how to run a simple (.tcl) program.
2. Visualized using network animator (nam).
3. Compared various routing protocols.