

Program 4:

Simulate the transmission of ping messages over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.

*program:*

```
set ns [ new Simulator ]
set nf [ open lab2.nam w ]
$ns namtrace-all $nf
set tf [ open lab2.tr w ]
$ns trace-all $tf
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$ns duplex-link $n0 $n4 1005Mb 1ms DropTail
$ns duplex-link $n1 $n4 50Mb 1ms DropTail
$ns duplex-link $n2 $n4 2000Mb 1ms DropTail
$ns duplex-link $n3 $n4 200Mb 1ms DropTail
$ns duplex-link $n4 $n5 1Mb 1ms DropTail
set p1 [new Agent/Ping] # letters A and P should be capital
$ns attach-agent $n0 $p1
$p1 set packetSize_ 50000
$p1 set interval_ 0.0001
set p2 [new Agent/Ping] # letters A and P should be capital
$ns attach-agent $n1 $p2
set p3 [new Agent/Ping] # letters A and P should be capital
$ns attach-agent $n2 $p3
$p3 set packetSize_ 30000
$p3 set interval_ 0.00001
set p4 [new Agent/Ping] # letters A and P should be capital
$ns attach-agent $n3 $p4
set p5 [new Agent/Ping] # letters A and P should be capital
$ns attach-agent $n5 $p5
$ns queue-limit $n0 $n4 5
$ns queue-limit $n2 $n4 3
$ns queue-limit $n4 $n5 2
Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts "node [$node_ id]received answer from $from with round trip time $rtt msec"
}
# please provide space between $node_ and id. No space between $ and from. No space between
and $ and rtt */
$ns connect $p1 $p5
$ns connect $p3 $p4
proc finish { } {
global ns nf tf
```

```
$ns flush-trace
close $nf
close $tf
exec nam lab2.nam &
exit 0
}
```

```
$ns at 0.1 "$p1 send"
$ns at 0.2 "$p1 send"
$ns at 0.3 "$p1 send"
$ns at 0.4 "$p1 send"
$ns at 0.5 "$p1 send"
$ns at 0.6 "$p1 send"
$ns at 0.7 "$p1 send"
$ns at 0.8 "$p1 send"
$ns at 0.9 "$p1 send"
```

```
$ns at 1.0 "$p1 send"
$ns at 1.1 "$p1 send"
$ns at 1.2 "$p1 send"
$ns at 1.3 "$p1 send"
$ns at 1.4 "$p1 send"
$ns at 1.5 "$p1 send"
$ns at 1.6 "$p1 send"
$ns at 1.7 "$p1 send"
$ns at 1.8 "$p1 send"
$ns at 1.9 "$p1 send"
$ns at 2.0 "$p1 send"
$ns at 2.1 "$p1 send"
$ns at 2.2 "$p1 send"
$ns at 2.3 "$p1 send"
$ns at 2.4 "$p1 send"
$ns at 2.5 "$p1 send"
$ns at 2.6 "$p1 send"
$ns at 2.7 "$p1 send"
$ns at 2.8 "$p1 send"
$ns at 2.9 "$p1 send"
$ns at 0.1 "$p3 send"
$ns at 0.2 "$p3 send"
$ns at 0.3 "$p3 send"
$ns at 0.4 "$p3 send"
$ns at 0.5 "$p3 send"
$ns at 0.6 "$p3 send"
$ns at 0.7 "$p3 send"
$ns at 0.8 "$p3 send"
$ns at 0.9 "$p3 send"
$ns at 1.0 "$p3 send"
$ns at 1.1 "$p3 send"
$ns at 1.2 "$p3 send"
$ns at 1.3 "$p3 send"
```

```
$ns at 1.4 "$p3 send"  
$ns at 1.5 "$p3 send"  
$ns at 1.6 "$p3 send"  
$ns at 1.7 "$p3 send"  
$ns at 1.8 "$p3 send"  
$ns at 1.9 "$p3 send"  
$ns at 2.0 "$p3 send"  
$ns at 2.1 "$p3 send"  
$ns at 2.2 "$p3 send"  
$ns at 2.3 "$p3 send"  
$ns at 2.4 "$p3 send"  
$ns at 2.5 "$p3 send"  
$ns at 2.6 "$p3 send"  
$ns at 2.7 "$p3 send"  
$ns at 2.8 "$p3 send"  
$ns at 2.9 "$p3 send"
```

```
$ns at 3.0 "finish"
```

```
$ns run
```

**AWK file:** (Open a new editor using “vi command” and write awk file and save with “.awk” extension)

```
BEGIN{  
drop=0;  
}  
{  
if($1=="d" )  
{  
drop++;  
}  
}  
END{  
printf("Total number of %s packets dropped due to congestion =%d\n",$5, drop);  
}
```

**Steps for execution**

1) Open vi editor and type program. Program name should have the extension “.tcl ”

```
[root@localhost ~]# vi lab2.tcl
```

2) Save the program by pressing “**ESC key**” first, followed by “**Shift and :**” keys simultaneously and type “**wq**” and press **Enter key**.

3) Open vi editor and type **awk** program. Program name should have the extension “.awk ”

```
[root@localhost ~]# vi lab2.awk
```

4) Save the program by pressing “**ESC key**” first, followed by “**Shift and :**” keys simultaneously and type “**wq**” and press **Enter key**.

5) Run the simulation program

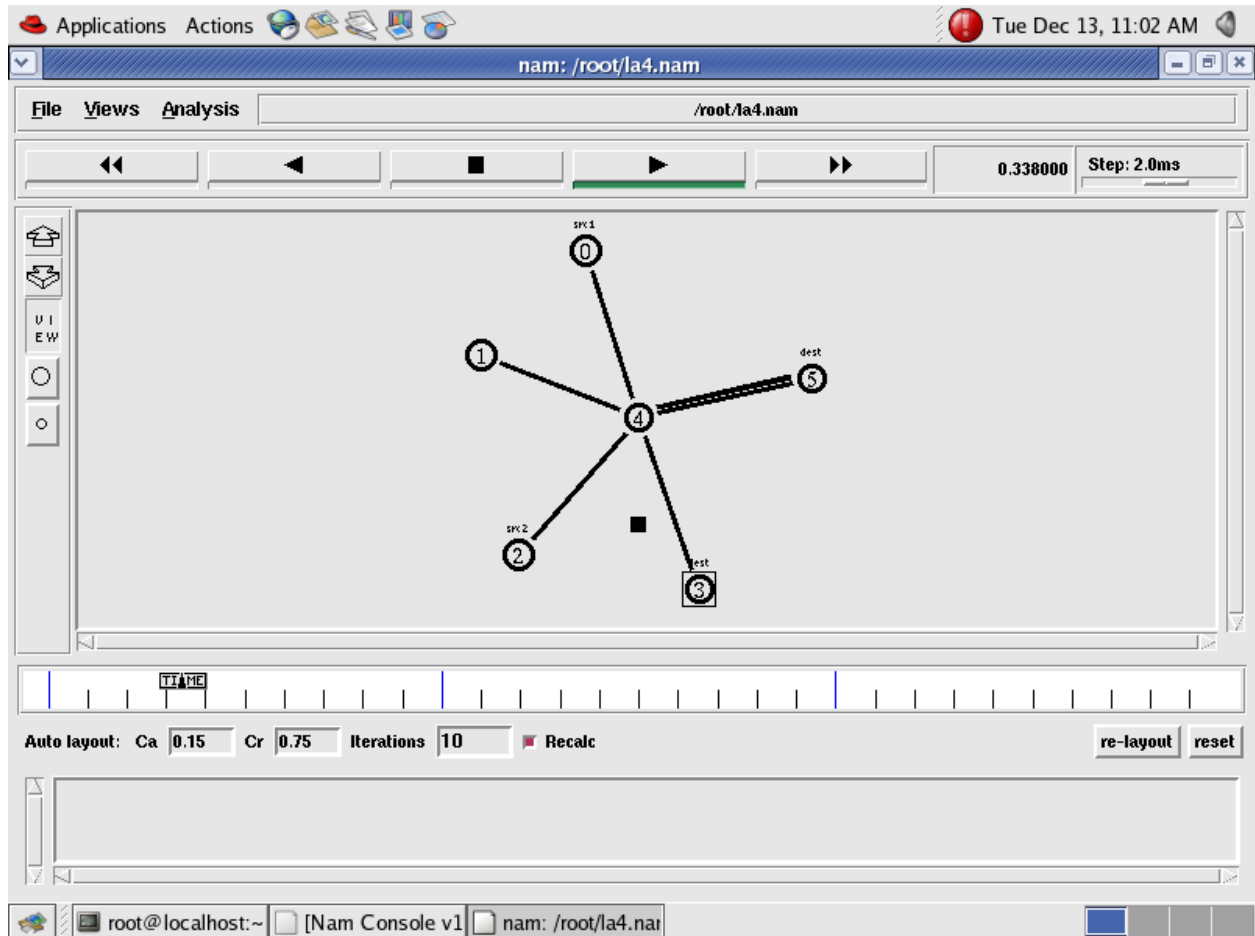
```
[root@localhost~]# ns lab2.tcl
```

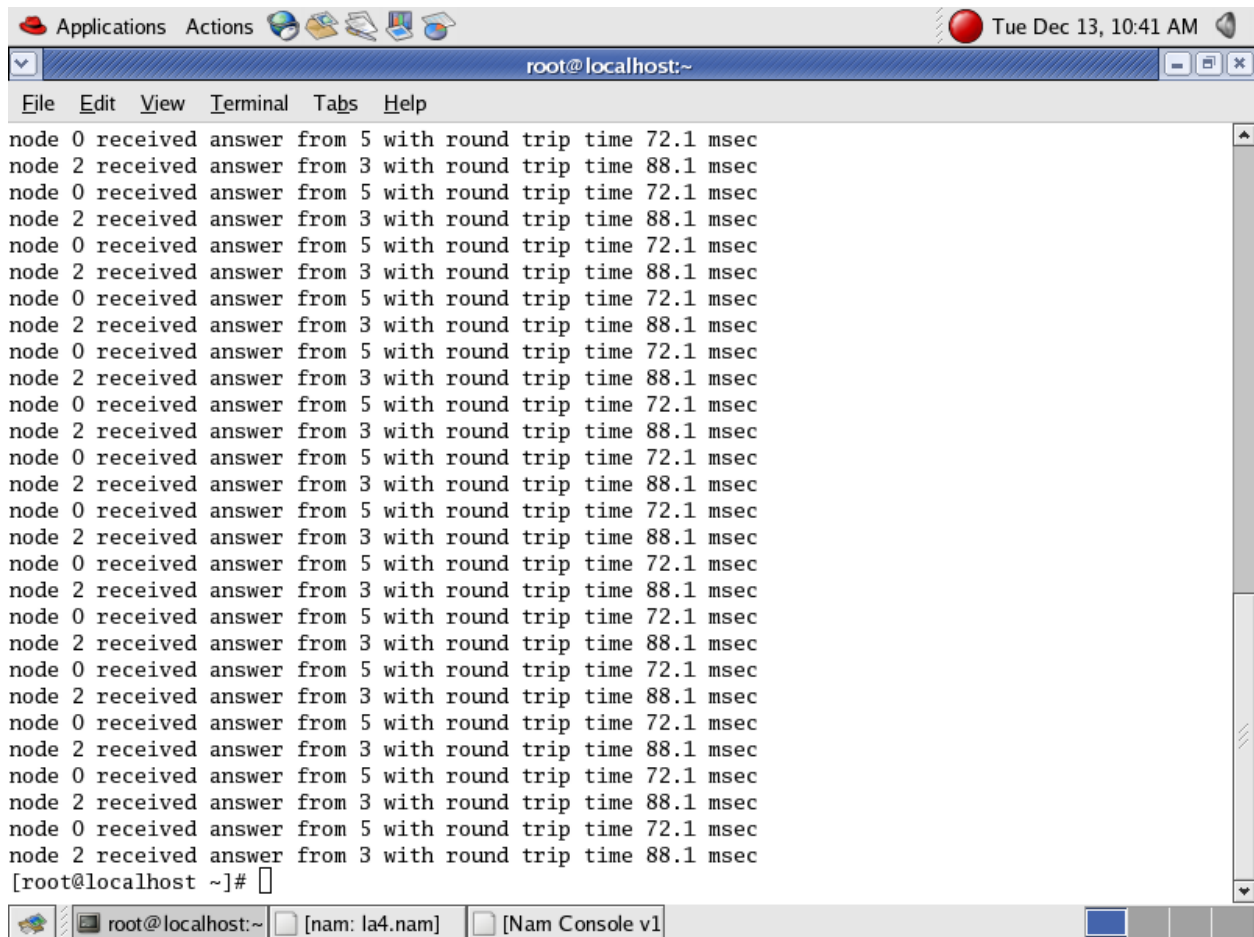
- i) Here “ns” indicates network simulator. We get the topology shown in the snapshot.
- ii) Now press the play button in the simulation window and the simulation will begins.
- 6) After simulation is completed run **awk file** to see the output ,

```
[root@localhost~]# awk -f lab2.awk lab2.tr
```

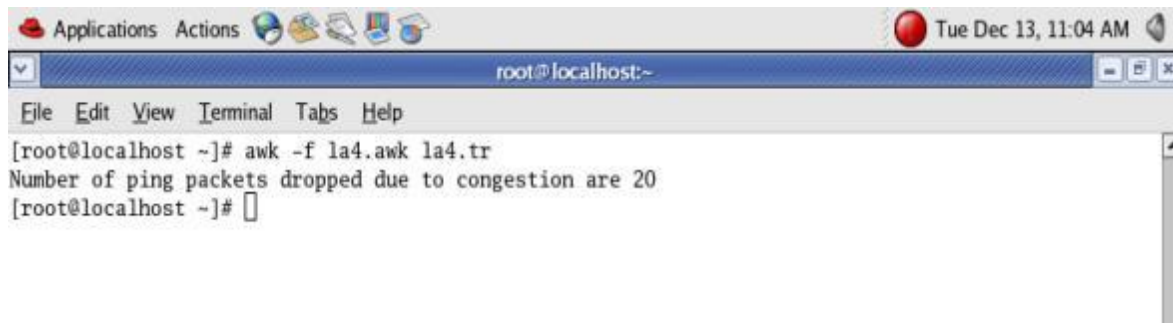
7) To see the trace file contents open the file as ,  
[root@localhost~]# vi lab2.tr

**Output:**



A terminal window titled 'root@localhost:~' with a menu bar (File, Edit, View, Terminal, Tabs, Help) and a status bar (root@localhost:~, [nam: la4.nam], [Nam Console v1]). The terminal displays a series of 24 lines of output from a network simulation, alternating between node 0 and node 2. Each line reports the receipt of an answer from either node 5 or node 3, along with the round trip time in milliseconds (72.1 msec or 88.1 msec). The output ends with a prompt '[root@localhost ~]#'.

```
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
[root@localhost ~]#
```

A terminal window titled 'root@localhost:~' with a menu bar (File, Edit, View, Terminal, Tabs, Help) and a status bar (root@localhost:~). The terminal shows the execution of the command 'awk -f la4.awk la4.tr', which outputs the message 'Number of ping packets dropped due to congestion are 20'. The prompt '[root@localhost ~]#' is visible at the end of the line.

```
[root@localhost ~]# awk -f la4.awk la4.tr
Number of ping packets dropped due to congestion are 20
[root@localhost ~]#
```

## Explanation for the code:

Certainly, let's go through the code line by line:

set ns [ new Simulator ]

- This line creates a new ns-2 simulator object and assigns it to the variable `ns`.

set nf [ open lab2.nam w ]

- Opens a file named "lab2.nam" for writing and assigns the file identifier to the variable `nf`. This file will be used for network animation.

```
$ns namtrace-all $nf
```

- Enables tracing of all events in the simulator and directs the output to the file specified by `nf` (lab2.nam).

```
set tf [ open lab2.tr w ]
```

- Opens a file named "lab2.tr" for writing and assigns the file identifier to the variable `tf`. This file will store the simulation trace.

```
$ns trace-all $tf
```

- Enables tracing of all events in the simulator and directs the output to the file specified by `tf` (lab2.tr).

```
set n0 [$ns node]  
set n1 [$ns node]  
set n2 [$ns node]  
set n3 [$ns node]  
set n4 [$ns node]  
set n5 [$ns node]
```

- Creates six nodes (`n0` to `n5`) in the network.

```
$ns duplex-link $n0 $n4 1005Mb 1ms DropTail  
$ns duplex-link $n1 $n4 50Mb 1ms DropTail  
$ns duplex-link $n2 $n4 2000Mb 1ms DropTail  
$ns duplex-link $n3 $n4 200Mb 1ms DropTail  
$ns duplex-link $n4 $n5 1Mb 1ms DropTail
```

- Sets up duplex links between nodes with specified characteristics (bandwidth, delay, and queuing discipline).

```
set p1 [new Agent/Ping]  
$ns attach-agent $n0 $p1  
$p1 set packetSize_ 50000  
$p1 set interval_ 0.0001
```

- Creates a Ping agent `p1`, attaches it to node `n0`, and configures its packet size and interval.

```
set p2 [new Agent/Ping]
$ns attach-agent $n1 $p2
```

- Creates another Ping agent `p2`, attaches it to node `n1`.

```
set p3 [new Agent/Ping]
$ns attach-agent $n2 $p3
$p3 set packetSize_ 30000
$p3 set interval_ 0.00001
```

- Creates a third Ping agent `p3`, attaches it to node `n2`, and configures its packet size and interval.

```
set p4 [new Agent/Ping]
$ns attach-agent $n3 $p4
```

- Creates another Ping agent `p4`, attaches it to node `n3`.

```
set p5 [new Agent/Ping]
$ns attach-agent $n5 $p5
```

- Creates the last Ping agent `p5` and attaches it to node `n5`.

```
$ns queue-limit $n0 $n4 5
$ns queue-limit $n2 $n4 3
$ns queue-limit $n4 $n5 2
```

- Sets queue limits for specific links.

```
Agent/Ping instproc recv {from rtt} {
    $self instvar node_
    puts "node [$node_ id] received answer from $from with round trip time $rtt msec"
}
```

- Defines a procedure named `recv` for the Ping agent to handle received responses. It prints information about the received packets, including the source node and round-trip time.

```
$ns connect $p1 $p5
$ns connect $p3 $p4
```

- Connects Ping agents to each other.

```

proc finish { } {
    global ns nf tf
    $ns flush-trace
    close $nf
    close $tf
    exec nam lab2.nam &
    exit 0
}

```

- Defines a procedure named `finish` to be executed when the simulation is complete. It flushes the trace, closes the animation and trace files, and then invokes the Network Animator (nam) to visualize the network animation.

```

$ns at 0.1 "$p1 send"
$ns at 0.2 "$p1 send"
$ns at 0.3 "$p1 send"
$ns at 0.4 "$p1 send"
$ns at 0.5 "$p1 send"
$ns at 0.6 "$p1 send"
$ns at 0.7 "$p1 send"
$ns at 0.8 "$p1 send"
$ns at 0.9 "$p1 send"
$ns at 1.0 "$p1 send"
$ns at 1.1 "$p1 send"
$ns at 1.2 "$p1 send"
$ns at 1.3 "$p1 send"
$ns at 1.4 "$p1 send"
$ns at 1.5 "$p1 send"
$ns at 1.6 "$p1 send"
$ns at 1.7 "$p1 send"
$ns at 1.8 "$p1 send"
$ns at 1.9 "$p1 send"
$ns at 2.0 "$p1 send"
$ns at 2.1 "$p1 send"
$ns at 2.2 "$p1 send"
$ns at 2.3 "$p1 send"
$ns at 2.4 "$p1 send"
$ns at 2.5 "$p1 send"
$ns at 2.6 "$p1 send"
$ns at 2.7 "$p1 send"
$ns at 2.8 "$p1 send"
$ns at 2.9 "$p1 send"
$ns at 0.1 "$p3 send"
$ns at 0.2 "$p3 send"
$ns at 0.3 "$p3 send"
$ns at 0.4 "$p3 send"

```



```
$ns at 0.5 "$p3 send"  
$ns at 0.6 "$p3 send"  
$ns at 0.7 "$p3 send"  
$ns at 0.8 "$p3 send"  
$ns at 0.9 "$p3 send"  
$ns at 1.0 "$p3 send"  
$ns at 1.1 "$p3 send"  
$ns at 1.2 "$p3 send"  
$ns at 1.3 "$p3 send"  
$ns at 1.4 "$p3 send"  
$ns at 1.5 "$p3 send"  
$ns at 1.6 "$p3 send"  
$ns at 1.7 "$p3 send"  
$ns at 1.8 "$p3 send"  
$ns at 1.9 "$p3 send"  
$ns at 2
```

```
.0 "$p3 send"  
$ns at 2.1 "$p3 send"  
$ns at 2.2 "$p3 send"  
$ns at 2.3 "$p3 send"  
$ns at 2.4 "$p3 send"  
$ns at 2.5 "$p3 send"  
$ns at 2.6 "$p3 send"  
$ns at 2.7 "$p3 send"  
$ns at 2.8 "$p3 send"  
$ns at 2.9 "$p3 send"  
$ns at 3.0 "finish"
```

- Schedules Ping messages to be sent at specific time intervals using `\$ns at`.

```
$ns run
```

- Initiates the simulation by running the ns-2 simulator.

The code sets up a network simulation with nodes exchanging Ping messages, configures various parameters, schedules events, and defines procedures to handle events and finish the simulation. The results are then visualized using the Network Animator (nam).