Program 1

Implement Three nodes point – to – point network with duplex links between them for different topologies. 1Set the queue size, vary the bandwidth, and find the number of packets dropped for various iterations.

```
set ns [new Simulator]        /* Letter S is capital */
set nf [open lab1.nam w] /* open a nam trace file in write mode */

$ns namtrace-all $nf           /* nf – nam file */

set tf [open lab1.tr w] /* tf- trace file */
$ns trace-all $tf

proc finish { } {        /* provide space b/w proc and finish and all are in small
case */global ns nf tf
$ns flush-trace        /* clears trace file
contents */close $nf
close $tf
exec nam
lab1.nam &
exit 0
}

set n0 [$ns node] /* creates 4
nodes */set n1 [$ns node]
set n2
[$ns
node]
set n3
[$ns
node]

$ns duplex-link $n0 $n2 200Mb 10ms DropTail /*Letter M is capital Mb*/
$ns duplex-link $n1 $n2 100Mb 5ms DropTail /*D and T are capital*/
$ns duplex-link $n2 $n3 1Mb 1000ms DropTail

$ns queue-limit $n0 $n2 10
$ns queue-limit $n1 $n2 10

set udp0 [new Agent/UDP] /* Letters A,U,D and P are capital */
$ns attach-agent $n0 $udp0

set cbr0 [new Application/Traffic/CBR] /* A,T,C,B and R are capital*/
$cbr0 set packetSize_ 500 /*S is capital, space after underscore*/
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
```

```
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1

set udp2 [new Agent/UDP]
$ns attach-agent $n2 $udp2

set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2

set null0 [new Agent/Null] /* A and N are capital */
$ns attach-agent $n3 $null0

$ns connect $udp0 $null0
$ns connect $udp1 $null0

$ns at 0.1 "$cbr0 start"
$ns at 0.2 "$cbr1 start"
$ns at 1.0 "finish"

$ns run
```

AWK file (Open a new editor using "vi command" and write awk file and save with ".awk" extension)

/*immediately after BEGIN should open braces '{'

```
BEGIN
{
C=0;
}
{
  If ($1=="d")
  {
    c++;
    printf("%s\t%s\n",$5,$11);
   }
 }
```

/*immediately after END should open braces '{'

```
END{
    printf("The number of packets dropped =%d\n",c);
}
```

Steps for execution

1) Open vi editor and type program. Program name should have the extension " **.tcl** "
   [root@localhost ~]# vi lab1.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 lab1.awk

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

5) Run the simulation program

[root@localhost~]# ns lab1.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 willbegins.

6) After simulation is completed run **awk file** to see the output ,

[root@localhost~]# awk –f lab1.awk lab1.tr

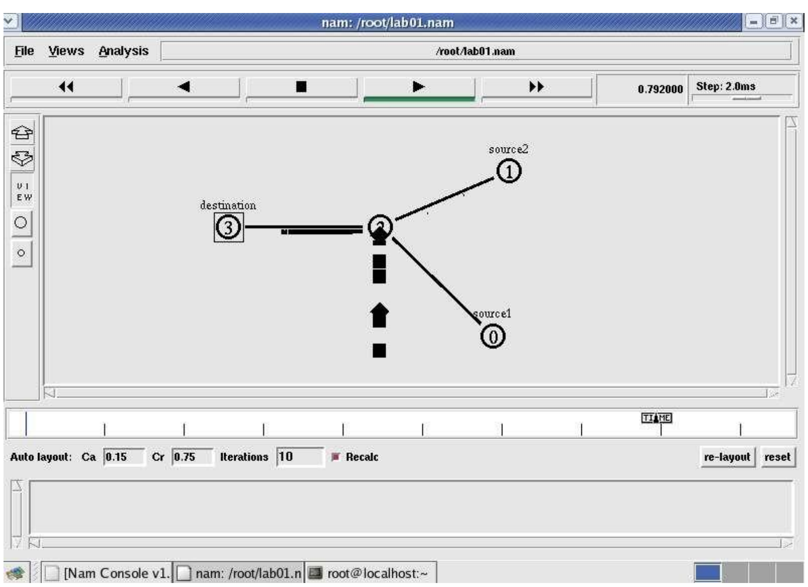
7) To see the trace file contents open the file as ,

[root@localhost~]# vi lab1.tr

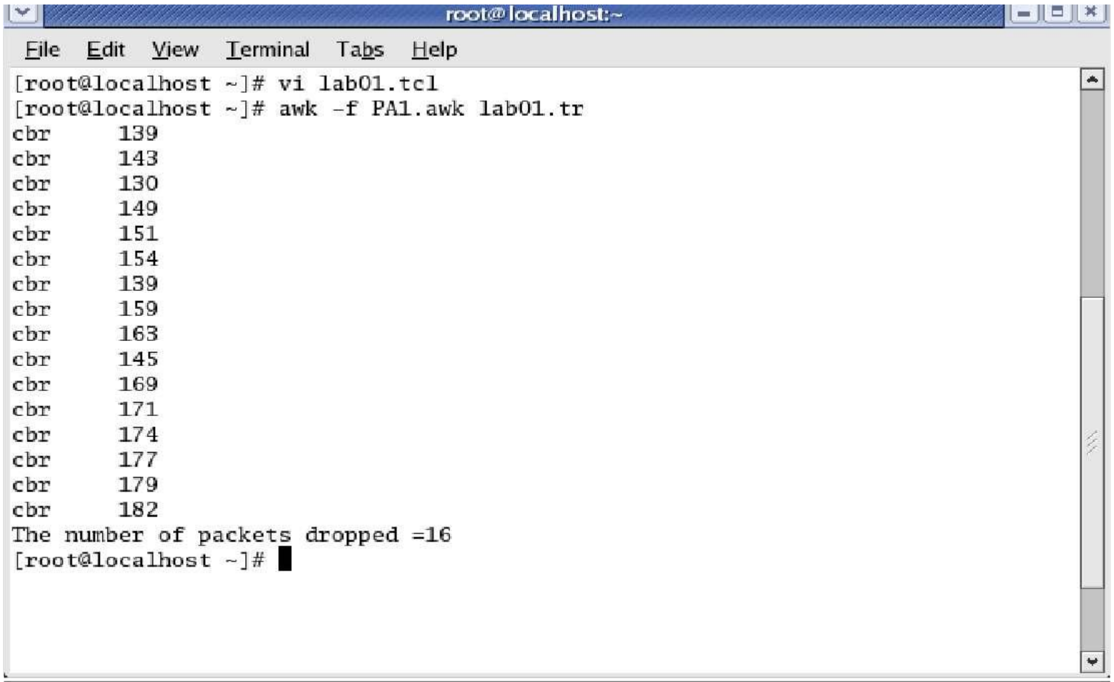
<u>Trace file contains 12 columns:-</u>

Event type, Event time, From Node, Source Node, Packet Type, Packet Size, Flags (indicated by------------------), Flow ID, Source address, Destination address, Sequence ID, Pocket ID,

Topology



Output



```
[root@localhost ~]# vi lab01.tcl
[root@localhost ~]# awk -f PA1.awk lab01.tr
cbr     139
cbr     143
cbr     130
cbr     149
cbr     151
cbr     154
cbr     139
cbr     159
cbr     163
cbr     145
cbr     169
cbr     171
cbr     174
cbr     177
cbr     179
cbr     182
The number of packets dropped =16
[root@localhost ~]#
```

Note:

1. Set the queue size fixed from n0 to n2 as 10, n1-n2 to 10 and from n2-n3 as 5.
   Syntax: To set the queue size
   $ns set queue-limit <from> <to> <size> Eg:
   $ns set queue-limit $n0 $n2 10
2. Go on varying the bandwidth from 10, 20 30 . . and find the number of packets dropped at the node

Explanation of the code:

Sl.no	Code	Explanation
1	set ns [new Simulator] /* Letter S is capital */	This line creates a new simulator object named `ns`.
	set nf [open lab1.nam w] /* open a nam trace file in write mode */	A new file named `lab1.nam` is opened in write mode, and the file handle is stored in the variable `nf`. This file will be used for NAM (Network Animator) trace.
3	$ns namtrace-all $nf /* nf – nam file */	This line enables NAM tracing for all events in the simulator (`$ns`) and directs the output to the file represented by the file handle `nf`.
4	set tf [open lab1.tr w] /* tf- trace file */	Similar to the NAM trace file, a

		new file named `lab1.tr` is opened in write mode, and the file handle is stored in the variable `tf`. This file will be used for general trace output.
5	$ns trace-all $tf	This line enables general tracing for all events in the simulator (`$ns`) and directs the output to the file represented by the file handle `tf`.
6	proc finish { } { /* provide space b/w proc and finish and all are in small case */ global ns nf tf $ns flush-trace/* clears trace file contents */ close $nf close $tf exec nam lab1.nam & exit 0 }	This block defines a procedure named `finish`. It flushes the traces, closes the NAM and general trace files, and then executes the NAM animator with the `lab1.nam` file. The `& exit 0` ensures that the script exits after executing NAM.
7	set n0 [$ns node] /* creates 4 nodes */ set n1 [$ns node] set n2 [$ns node] set n3 [$ns node]	These lines create four nodes (`n0`, `n1`, `n2`, and `n3`) in the network simulation.
8	$ns duplex-link $n0 $n2 200Mb 10ms DropTail /*Letter M is capital Mb*/ $ns duplex-link $n1 $n2 100Mb 5ms DropTail /*D and T are capital*/ $ns duplex-link $n2 $n3 1Mb 1000ms DropTail	These lines create duplex links between nodes with specified bandwidth, delay, and queuing mechanism. The third and fourth arguments represent bandwidth and delay in each case.
9	$ns queue-limit $n0 $n2 10 $ns queue-limit $n1 $n2 10	These lines set the queue limit for the links between nodes.
	set udp0 [new Agent/UDP] /* Letters A,U,D and P are capital */ $ns attach-agent $n0 $udp0	Here, an Agent/UDP object named `udp0` is created, and it is attached to node `n0`.
	set cbr0 [new Application/Traffic/CBR] /* A,T,C,B and R are capital*/ $cbr0 set packetSize_ 500 /*S is capital, space after underscore*/ $cbr0 set interval_ 0.005 $cbr0 attach-agent $udp0	This block creates a CBR traffic source (`cbr0`) attached to `udp0`, and sets its packet size and interval.
	$ns connect $udp0 $null0 $ns connect $udp1 $null0	These lines connect the UDP agents to a Null agent, effectively directing their traffic to a "black hole."

	$ns at 0.1 "$cbr0 start" $ns at 0.2 "$cbr1 start" $ns at 1.0 "finish"	These lines schedule the start of CBR traffic from `cbr0` and `cbr1` at simulation time 0.1 and 0.2, respectively. The `finish` procedure is scheduled to run at simulation time 1.0.
	$ns run	This line starts the simulation.
	AWK file	

1	BEGIN
	{
	C=0;
	}
		This block is the BEGIN block in AWK, which is executed before processing any lines from the input. In this block, a variable `C` is initialized to 0.

2	{
	if ($1 == "d")
	{
	C++;
	printf("%s\t%s\n", $5, $11);
	}
	}
		This block is executed for each line in the input. It checks if the first field (`$1`) is equal to "d". If true, it increments the variable `C` by 1 and prints the 5th and 11th fields separated by a tab.

3	END
	{
	printf("The number of packets dropped = %d\n", C);
	}
		This block is the END block in AWK, which is executed after processing all lines from the input. It prints a message indicating the number of packets dropped, using the value stored in the variable `C`.

AWK file:

AWK, a text processing and pattern scanning language.

Logic Explanation:

1. The `BEGIN` block initializes a counter variable `C` to 0.

2. The main block checks each line of input. If the first field is "d", it increments the counter `C` and prints the 5th and 11th fields.

3. The `END` block prints the total number of packets dropped based on the value of the counter `C`.