THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Name: Tanvi Khasgiwal
Roll No: 42

Date:22/02/2021

Lab Assignment No: 4

Aim: Implementation of Specific Network topology with respect to UDP

Lab Outcome Attained: Executing the code in tcl file with respect to UDP connection.

Theory:

UDP uses a simple connectionless communication model with a minimum of protocol mechanisms. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram. It has no handshaking dialogues, and thus exposes the user's program to any unreliability of the underlying network; there is no guarantee of delivery, ordering, or duplicate protection. If error-correction facilities are needed at the network interface level, an application may use Transmission Control Protocol (TCP) or Stream Control Transmission Protocol (SCTP) which are designed for this purpose.

UDP is suitable for purposes where error checking and correction are either not necessary or are performed in the application; UDP avoids the overhead of such processing in the protocol stack. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for packets delayed due to retransmission, which may not be an option in a real-time system.

Code and its explanation:

set ns [new Simulator]

Creating the Simulator object and assign it name “ns” A simulation in ns is described by a Tcl class Simulator. A simulation script, therefore, generally begins by creating an instance of this class. This is done with the command.*

```
$ns color 1 Blue

$ns color 2 Red
```

set color of the packets for a flow specified by the flow id (fid). This member function of "Simulator" object is for the NAM display, and has no effect on the actual simulation.*

```
set nf [open tanvi.nam w]

$ns namtrace-all $nf

set np [open tanvi.tr w]

$ns trace-all $np
```

The output is two trace files, out.tr and out.nam. When the simulation completes at the end, it will attempt to run a nam visualisation of the simulation on your screen. This member function tells the simulator to record simulation traces in NAM input format. Similarly, the member function trace-all is for recording the simulation trace in a general format. It first opens the file 'out.nam' for writing and gives it the file handle 'nf'. It then tells the simulator to trace each packet on every link in the topology and write the data into the file.*

```
#Define a 'finish' procedure

proc finish {} {

        global ns nf np

        $ns flush-trace

        #Close the NAM trace file

        close $nf

        #Execute NAM on the trace file

        exit 0

}
```

Proc finish is called after this simulation is over by the command $ns at 5.0 "finish". In this function, post-simulation processes are specified.*

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

To create a node we can simply use the simulator method node. The above four lines create four nodes and assign them to the handles 'n0' ,'n1','n2,'n3''*

$ns duplex-link $n0 $n2 2Mb 10ms DropTail

$ns duplex-link $n1 $n2 2Mb 10ms DropTail

$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

We can then either use the simulator method simplex-link or the method duplex-link to connect the nodes with a link. Create a new bidirectional link between node1 and node2 with bandwidth bw in bits per second and link propagation delay delay in seconds. In the above simulation script, DropTail queue is used. In addition to the DropTail queue, ns also supports many other queueing policies like for example FairQueueing(FQ), Random Early Detection (RED), and Class-based queueing (CBQ). *

$ns queue-limit $n2 $n3 10

Set the maximum number of packets that can be queued on the link in the direction from node1 to node2 to queue-limit. The link between node1 and node2 should have already been created.*

$ns duplex-link-op $n0 $n2 orient right-down

$ns duplex-link-op $n1 $n2 orient right-up

$ns duplex-link-op $n2 $n3 orient right

Give node position (for NAM). This command is used to set link attributes (orientation) for duplex links. It defines the orientation of the links between the nodes*

$ns duplex-link-op $n2 $n3 queuePos 0.25

The line added to the above code id to monitor the queue for the link from n2 to n3. 0.25 will result in an angle of 45 degree*

set udp [new Agent/UDP]

This creates an instance of the UDP agent.*

$ns attach-agent $n1 $udp

This is a common command used to attach any <agent> to a given <node>.UDP Agent is connected to n1.*

set null [new Agent/Null]

*** Creating a receveing agent i.e a null agent*.**

$ns attach-agent $n3 $null

Null agent is connected to n3.*

$ns connect $udp $null

UDP Source and destination is connected*

$udp set fid_ 2

#Setup a CBR over UDP connection Create a CBR traffic source

set cbr [new Application/Traffic/CBR]

A CBR traffic generator agent is simulated.*

$cbr attach-agent $udp

attached to the UDP agent*

$cbr set type_ CBR

$cbr set packet_size_ 1000

Over here the Packet size in bytes and the interval in seconds, 1000 bits/second.*

$cbr set rate_ 1mb

It defines the rate of transfer. *

$cbr set random_ false

With the random_ parameter it is possible to add some randomness in the interarrival times of the packets. *

$ns at 0.1 "$cbr start"

$ns at 4.5 "$cbr stop"

This member function of a Simulator object makes the scheduler (scheduler_ is the variable that points the scheduler object created by [new Scheduler] command at the beginning of the script) to schedule the execution of the specified string at given simulation time. For example, $ns at 0.1 "$cbr start" will make the scheduler call a start member function of the CBR traffic source object, which starts the CBR to transmit data. In NS, usually a traffic source does not transmit actual data, but it notifies the underlying agent that it has some amount of data to transmit, and the agent, just knowing how much of the data to transfer, creates packets and sends them.*

$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

****Detach tcp and sink agents (not really necessary)****

$ns at 5.0 "finish"

****It calls the finish procedure after 5 seconds of simulation time****

$ns run

****It now saves the file and runs the simulation****

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

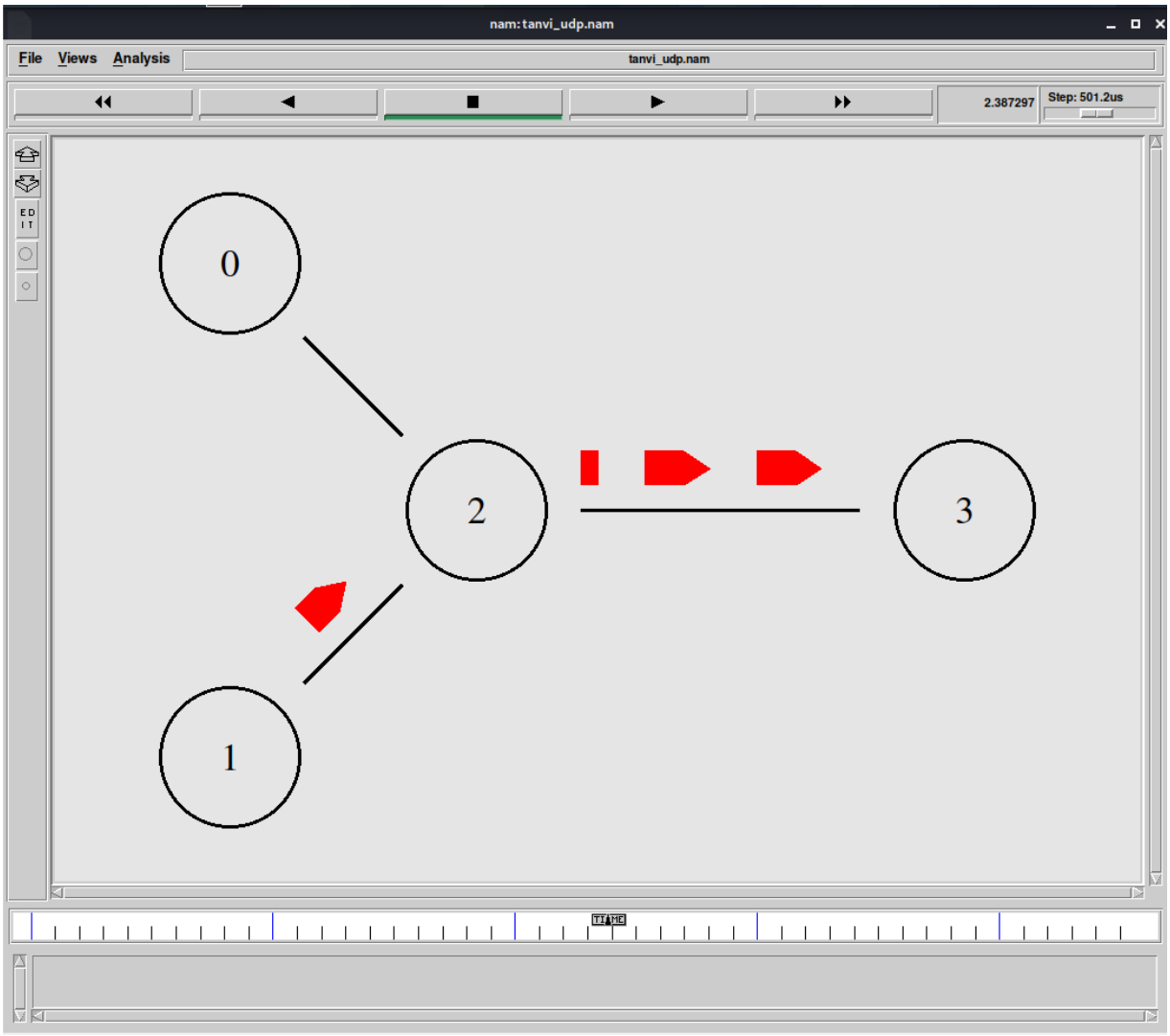
Name: Tanvi Khasgiwal
Roll No: 42

Screenshots:

```
tanvi@Tanvi: ~
File  Actions  Edit  View  Help
┌──(tanvi㉿Tanvi)-[~]
└─$ ls
 b1new.tcl.txt   Documents   index.html     Music      'routing (copy 1).tcl'   tanvi.tcl       Videos
 b1.tcl          Downloads   index.html.1   Pictures    routing_new.tcl         tanvi_udp.tcl
 Desktop         hello.tcl   index.html.2   Public      routing.tcl             Templates

┌──(tanvi㉿Tanvi)-[~]
└─$ ns tanvi_udp.tcl
CBR packet size = 1000
CBR interval = 0.0080000000000000002

┌──(tanvi㉿Tanvi)-[~]
└─$ nam tanvi_udp.nam
```



THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Name: Tanvi Khasgiwal
Roll No: 42

/home/tanvi/tanvi_udp.tr - Mousepad

File Edit Search View Document Help

```
+ 0.1 1 2 cbr 1000 ------- 2 1.0 3.0 0 0
- 0.1 1 2 cbr 1000 ------- 2 1.0 3.0 0 0
+ 0.108 1 2 cbr 1000 ------- 2 1.0 3.0 1 1
- 0.108 1 2 cbr 1000 ------- 2 1.0 3.0 1 1
r 0.114 1 2 cbr 1000 ------- 2 1.0 3.0 0 0
+ 0.114 2 3 cbr 1000 ------- 2 1.0 3.0 0 0
- 0.114 2 3 cbr 1000 ------- 2 1.0 3.0 0 0
+ 0.116 1 2 cbr 1000 ------- 2 1.0 3.0 2 2
- 0.116 1 2 cbr 1000 ------- 2 1.0 3.0 2 2
r 0.122 1 2 cbr 1000 ------- 2 1.0 3.0 1 1
+ 0.122 2 3 cbr 1000 ------- 2 1.0 3.0 1 1
- 0.122 2 3 cbr 1000 ------- 2 1.0 3.0 1 1
+ 0.124 1 2 cbr 1000 ------- 2 1.0 3.0 3 3
- 0.124 1 2 cbr 1000 ------- 2 1.0 3.0 3 3
r 0.13 1 2 cbr 1000 ------- 2 1.0 3.0 2 2
+ 0.13 2 3 cbr 1000 ------- 2 1.0 3.0 2 2
- 0.13 2 3 cbr 1000 ------- 2 1.0 3.0 2 2
+ 0.132 1 2 cbr 1000 ------- 2 1.0 3.0 4 4
- 0.132 1 2 cbr 1000 ------- 2 1.0 3.0 4 4
r 0.138 1 2 cbr 1000 ------- 2 1.0 3.0 3 3
+ 0.138 2 3 cbr 1000 ------- 2 1.0 3.0 3 3
- 0.138 2 3 cbr 1000 ------- 2 1.0 3.0 3 3
r 0.138706 2 3 cbr 1000 ------- 2 1.0 3.0 0 0
+ 0.14 1 2 cbr 1000 ------- 2 1.0 3.0 5 5
- 0.14 1 2 cbr 1000 ------- 2 1.0 3.0 5 5
r 0.146 1 2 cbr 1000 ------- 2 1.0 3.0 4 4
+ 0.146 2 3 cbr 1000 ------- 2 1.0 3.0 4 4
- 0.146 2 3 cbr 1000 ------- 2 1.0 3.0 4 4
r 0.146706 2 3 cbr 1000 ------- 2 1.0 3.0 1 1
+ 0.148 1 2 cbr 1000 ------- 2 1.0 3.0 6 6
- 0.148 1 2 cbr 1000 ------- 2 1.0 3.0 6 6
r 0.154 1 2 cbr 1000 ------- 2 1.0 3.0 5 5
+ 0.154 2 3 cbr 1000 ------- 2 1.0 3.0 5 5
- 0.154 2 3 cbr 1000 ------- 2 1.0 3.0 5 5
r 0.154706 2 3 cbr 1000 ------- 2 1.0 3.0 2 2
+ 0.156 1 2 cbr 1000 ------- 2 1.0 3.0 7 7
- 0.156 1 2 cbr 1000 ------- 2 1.0 3.0 7 7
r 0.162 1 2 cbr 1000 ------- 2 1.0 3.0 6 6
+ 0.162 2 3 cbr 1000 ------- 2 1.0 3.0 6 6
- 0.162 2 3 cbr 1000 ------- 2 1.0 3.0 6 6
r 0.162706 2 3 cbr 1000 ------- 2 1.0 3.0 3 3
+ 0.164 1 2 cbr 1000 ------- 2 1.0 3.0 8 8
- 0.164 1 2 cbr 1000 ------- 2 1.0 3.0 8 8
r 0.17 1 2 cbr 1000 ------- 2 1.0 3.0 7 7
+ 0.17 2 3 cbr 1000 ------- 2 1.0 3.0 7 7
- 0.17 2 3 cbr 1000 ------- 2 1.0 3.0 7 7
r 0.170706 2 3 cbr 1000 ------- 2 1.0 3.0 4 4
```

Conclusion: Successful implementation of Specific Network topology with respect to UDP.