Experiment no. 10

SEMESTER: V(2016-17) **DATE OF DECLARATION:** 3/10/16

SUBJECT: CN **DATE OF SUBMISSION:** 10/10/16

NAME OF THE STUDENT:FAISAL KHAN **ROLL NO.: 32**

Aim	To build a topology using ns2.
Learning Objective	The student will use ns2 to build topology.
Learning Outcome	The student will create the basic topology and analyze it with respect to the different parameters such as the 'packetsize_' and 'interval_'.
Course Outcome	C304.1: Conceptualize all the OSI and TCP/IP layers C304.4: Select appropriate network tools to build network topologies.
Program Outcome	PO1: Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization for the solution of complex engineering problem. PO5: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools, including prediction and modelling to complex engineering activities, with an understanding of the limitations. PO9: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
Bloom's Taxonomy Level	Remember Apply Analyze Create
Theory	To simulate a simple topology first develop a Tcl script for ns2. First learn how to set up nodes and links, how to send data from one node to another, how to monitor a queue and how to start nam from your simulation script to visualize simulation. **1. How to start**

Now write a 'template' that you will use to create first Tcl scripts. Write Tcl scripts in any text editor like notepad and call it as 'example1.tcl'.

First of all, you create a simulator object. This is done with the command.

```
set ns [new Simulator]
```

Now open a file for writing that is going to be used for the nam trace data.

```
set nf [open out.nam w]
$ns namtrace-all $nf
```

The first line opens the file 'out.nam' for writing and gives it the file handle 'nf'. The second line tells the simulator object that we created above to write all simulation data that is going to be relevant for nam into this file.

The next step is to add a 'finish' procedure that closes the trace file and starts nam.

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

The next line tells the simulator object to execute the 'finish' procedure after 5.0 seconds of simulation time.

```
$ns at 5.0 "finish"
```

ns provides you with a very simple way to schedule events with the 'at' command. The last line finally starts the simulation.

```
$ns run
```

2. Two nodes, one link

This section define a very simple topology with two nodes that are connected by a link. The following two lines define the two nodes. (Note: Insert the code in this section **before** the line '$ns run', or even better, before the line '$ns at 5.0 "finish"').

```
set n0 [$ns node]
```

```
set n1 [$ns node]
```

A new node object is created with the command '$ns node'. The above code creates two nodes and assigns them to the handles 'n0' and 'n1'. The next line connects the two nodes.

```
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

This line tells the simulator object to connect the nodes n0 and n1 with a duplex link with the bandwidth 1Megabit, a delay of 10ms and a DropTail queue. Now save your file and start the script with 'ns example1.tcl'. nam will be started automatically and see an output that resembles the picture below.

3. Sending data

Of course, this example isn't very satisfying yet, since you can only look at the topology, but nothing actually happens, so the next step is to send some data from node n0 to node n1. In ns, data is always being sent from one 'agent' to another. So the next step is to create an agent object that sends data from node n0, and another agent object that receives the data on node n1.

```
#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

These lines create a UDP agent and attach it to the node n0, then attach a CBR traffic generatot to the UDP agent. CBR stands for 'constant bit rate'. Line 7 and 8 should be self-explaining. The packetSize is being set to 500 bytes and a packet will be sent every 0.005 seconds (i.e. 200 packets per second).

The next lines create a Null agent which acts as traffic sink and attach it to node n1.

```
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
```

Now the two agents have to be connected with each other.

```
$ns connect $udp0 $null0
```

And now we have to tell the CBR agent when to send data and when to stop sending. Note: It's probably best to put the following lines just before the line '$ns at 5.0 "finish"'.

```
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

Now you can save the file and start the simulation again. When you click on the 'play' button in the nam window, you will see that after 0.5 simulation seconds, node 0 starts sending data packets to node 1. You might want to slow nam down then with the 'Step' slider.

4. As always, the first step is to define the topology. You should create a file 'example2.tcl'. You will always have to create a simulator object, you will always have to start the simulation with the same command, and if you want to run nam automatically, you will always have to open a trace file, initialize it, and define a procedure which closes it and starts nam.
Now insert the following lines into the code to create four nodes.

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

The following piece of Tcl code creates three duplex links between the nodes.

```
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms DropTail
```

You can save and start the script now. You might notice that the topology looks a bit awkward in nam. You can hit the 're-layout' button to make it look better, but it would be nice to have some more control over the layout. Add the next three lines to your Tcl script and start it again.

```
$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
```

You will probably understand what this code does when you look at the topology in the nam window now. It should look like the picture below.

Note that the autolayout related parts of nam are gone, since now you have taken the layout into your own hands. The options for the orientation of a link

are right, left, up, down and combinations of these orientations. You can experiment with these settings later, but for now please leave the topology the way it is.

5. The events

Now we create two UDP agents with CBR traffic sources and attach them to the nodes n0 and n1. Then we create a Null agent and attach it to node n3.

```
#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#Create a UDP agent and attach it to node n1
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1

# Create a CBR traffic source and attach it to udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
```

The two CBR agents have to be connected to the Null agent.

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

We want the first CBR agent to start sending at 0.5 seconds and to stop at 4.5 seconds while the second CBR agent starts at 1.0 seconds and stops at 4.0 seconds.

```
$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$cbr1 start"
$ns at 4.0 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"
```

When you start the script now with 'ns example2.tcl', you will notice that there is more traffic on the links from n0 to n2 and n1 to n2 than the link from n2 to n3 can carry. A simple calculation confirms this: We are sending 200 packets per second on each of the first two links and the packet size is 500 bytes. This results in a bandwidth of 0.8 megabits per second for the

links from n0 to n2 and from n1 to n2. That's a total bandwidth of 1.6Mb/s, but the link between n2 and n3 only has a capacity of 1Mb/s, so obviously some packets are being discarded. But which ones? Both flows are black, so the only way to find out what is happening to the packets is to monitor them in nam by clicking on them. In the next two sections I'm going to show you how to distinguish between different flows and how to see what is actually going on in the queue at the link from n2 to n3.

6. Marking flows

Add the following two lines to your CBR agent definitions.

```
$udp0 set class_ 1
$udp1 set class_ 2
```

The parameter 'fid_' stands for 'flow id'.
Now add the following piece of code to your Tcl script, preferably at the beginning after the simulator object has been created, since this is a part of the simulator setup.

```
$ns color 1 Blue
$ns color 2 Red
```

This code allows you to set different colors for each flow id.

Now you can start the script again and one flow should be blue, while the other one is red. Watch the link from node n2 to n3 for a while, and you will notice that after some time the distribution between blue and red packets isn't too fair anymore (at least that's the way it is on my system). In the next section I'll show you how you can look inside this link's queue to find out what is going on there.

7. Monitoring a queue

You only have to add the following line to your code to monitor the queue for the link from n2 to n3.

```
$ns duplex-link-op $n2 $n3 queuePos 0.5
```

Start ns again and you will see a picture similar to the one below after a few moments.

You can see the packets in the queue now, and after a while you can even see how the packets are being dropped, though (at least on my system, I guess it might be different in later or earlier releases) only blue packets are being

dropped. But you can't really expect too much 'fairness' from a simple DropTail queue. So let's try to improve the queueing by using a SFQ (stochastic fair queueing) queue for the link from n2 to n3. Change the link definition for the link between n2 and n3 to the following line.

$ns duplex-link $n3 $n2 1Mb 10ms SFQ

The queueing should be 'fair' now. The same amount of blue and red packets should be dropped.

Lab Activities

1. Form a group of two students.

Name of the Group	Roll No.	Name of the student

2. Select a topology of your choice(Note: not the above one)
3. Attach the code for topology.
4. Attach the snapshot of the topology here with explanation.
5. Start the simulation and attach the snapshot of the simulation.
6. Click on the packet and attach the snapshot of it.
7. Click on the link and attach the snaphot of it.
8. Now again start the simulation and see the dropping packets. Take a snapshot and attach it.
9. Change the 'packetsize_' and 'interval_' parameters in your code. Attach the code.
10. Start the simulation. Take a snapshot of this simualtion.
11. Click on the packet and attach the snapshot of it.
12. Click on the link and attach the snpshot it.
13. Write your observation.(Note: while writing observation refer to the observation given in section 5)

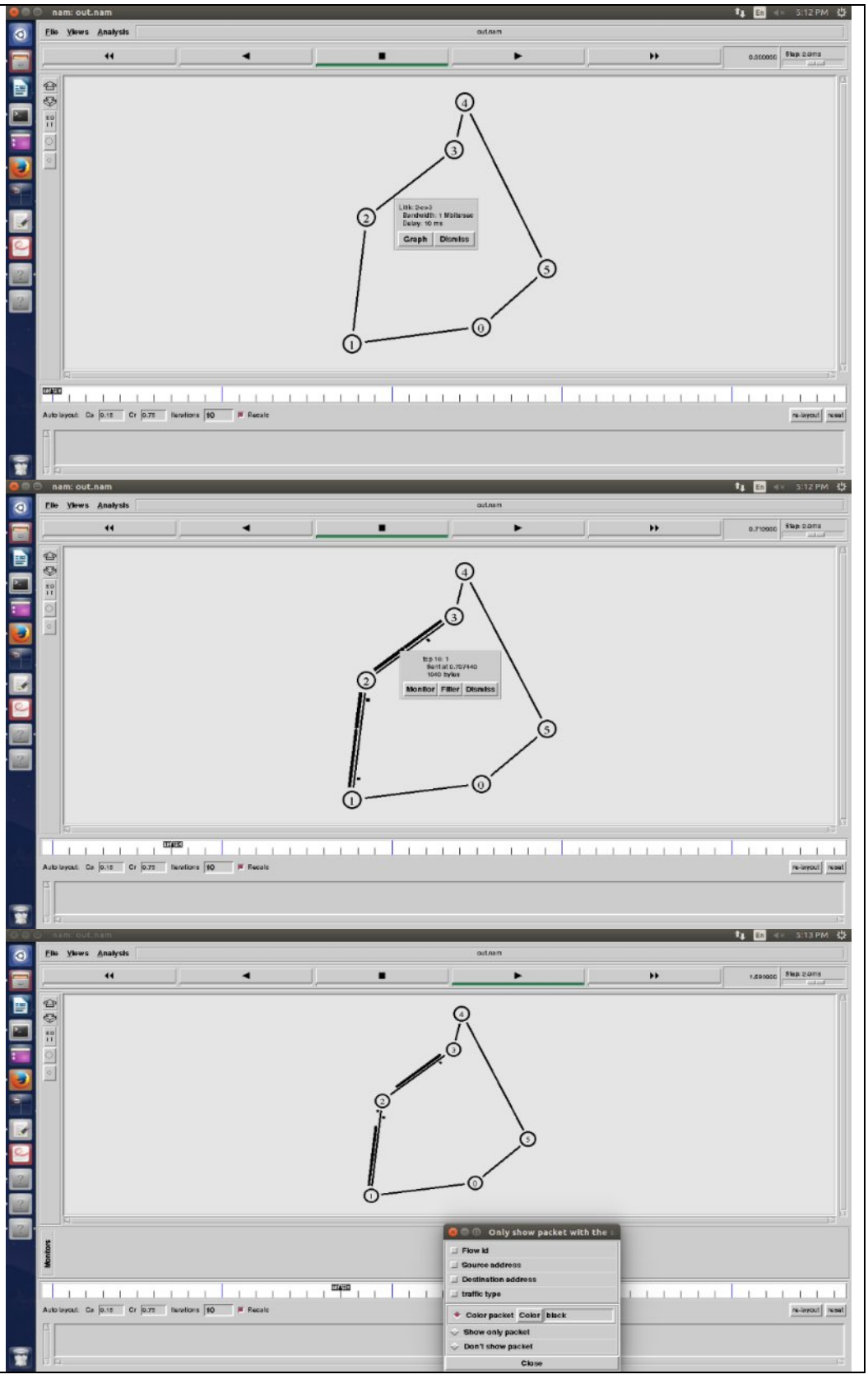
Ans.)

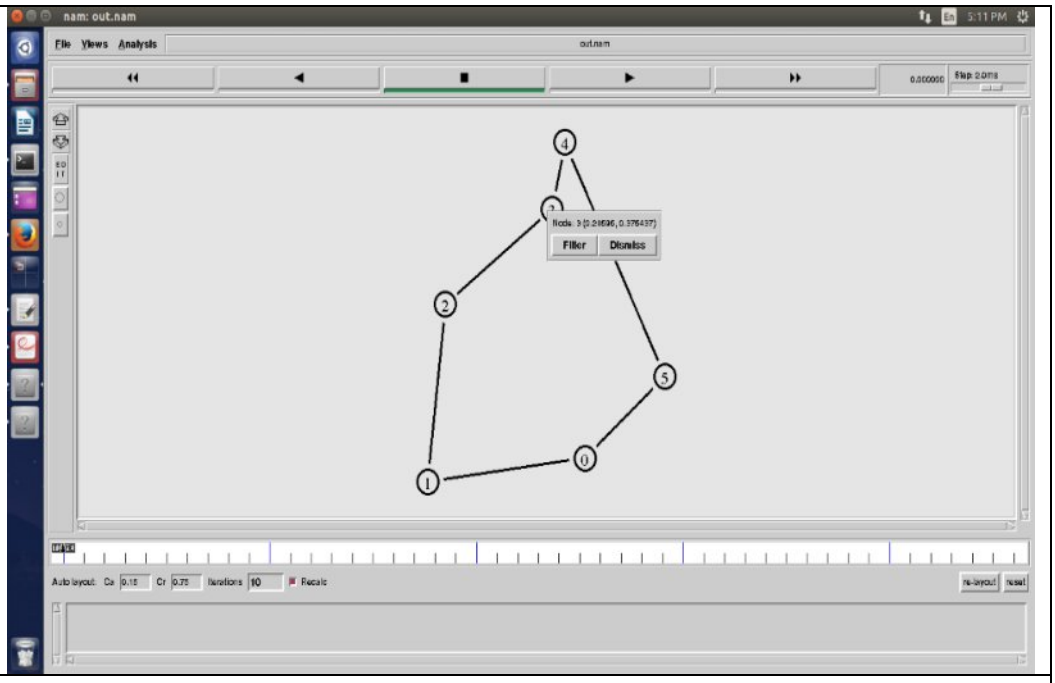
Ring Toplogy:

```
set ns [new Simulator]

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

#Define a 'finish' procedure
proc finish {} {
```

```
    global ns nf
    $ns flush-trace
    #Close the trace file
    close $nf
    #Executenam on the trace file
    exec nam out.nam &
    exit0
}


#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]


#Change the shape of center node in a star topology
$n0 shape square

#Create links between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n0 $n3 1Mb 10ms DropTail
$ns duplex-link $n0 $n4 1Mb 10ms DropTail
$ns duplex-link $n0 $n5 1Mb 10ms DropTail


#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$ns attach-agent $n1 $tcp0
#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node
 n3
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
#Connect the traffic sources with the traffic sink
$ns connect $tcp0 $sink0


# Create a CBR traffic source and attach it to tcp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0

#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"


#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

O/P:


nam: out.nam
File Views Analysis
out.nam
Graph Dismiss
Monitor Filter Dismiss
Only show packet with the
Flow Id
Source address
Destination address
traffic type
Color packet Color black
Show only packet
Don't show packet
Close
Monitors

	
References	1. B.A. Forouzan, “Data Communications and Networking”, TMH, Fourth Edition. 2. http://www.isi.edu/nsnam/ns/tutorial/