LIST OF EXPERIMENTS

LAB 1: Packet Sniffing and Wireshark

LAB 2: Introduction to NS2 and TCL scripting

LAB 3: Simulate a three-node point-to-point network with a duplex link between them for the following cases:

a. One node act as the sender and receiver and the remaining nodes acts as a repeater
b. One node act as the sender and two nodes acts receiver and data should flow in two different directions
c. Two nodes acts as sender and one node acts as receiver. Set queue size to 3 and vary the bandwidth (two cases) and count the packets being dropped

NOTE: Use TCP as transport layer protocol and FTP as traffic source for all cases

Constraint: In all the above cases, no node or link should be kept inactive at any instant of time

LAB 4: Simulate a four-node point-to-point network and connect the link as follows: Apply TCP agent between n0 and n3. Apply relevant applications over TCP and UDP. Change bandwidth and latency (two cases) and determine the number of packets sent by two agents

LAB 5: A) Simulate the Link State Routing Protocol for the topology and write the appropriate conclusion and deduce the relevant inference from your observations

B) Simulate the Distance Vector Routing Protocol for the topology and write the appropriate conclusion and deduce the relevant inference from your observations

LAB 6: Simulate a 6 node Ethernet LAN network Change Error rate and Data rate and compare the throughput. Note: Have two cases of different data and error rates

LAB 7: Simulation of STOP and WAIT Protocol

LAB 8: NETSIM experiments- I

LAB 9: NETSIM experiments- II

LAB 10: NETSIM experiments- III

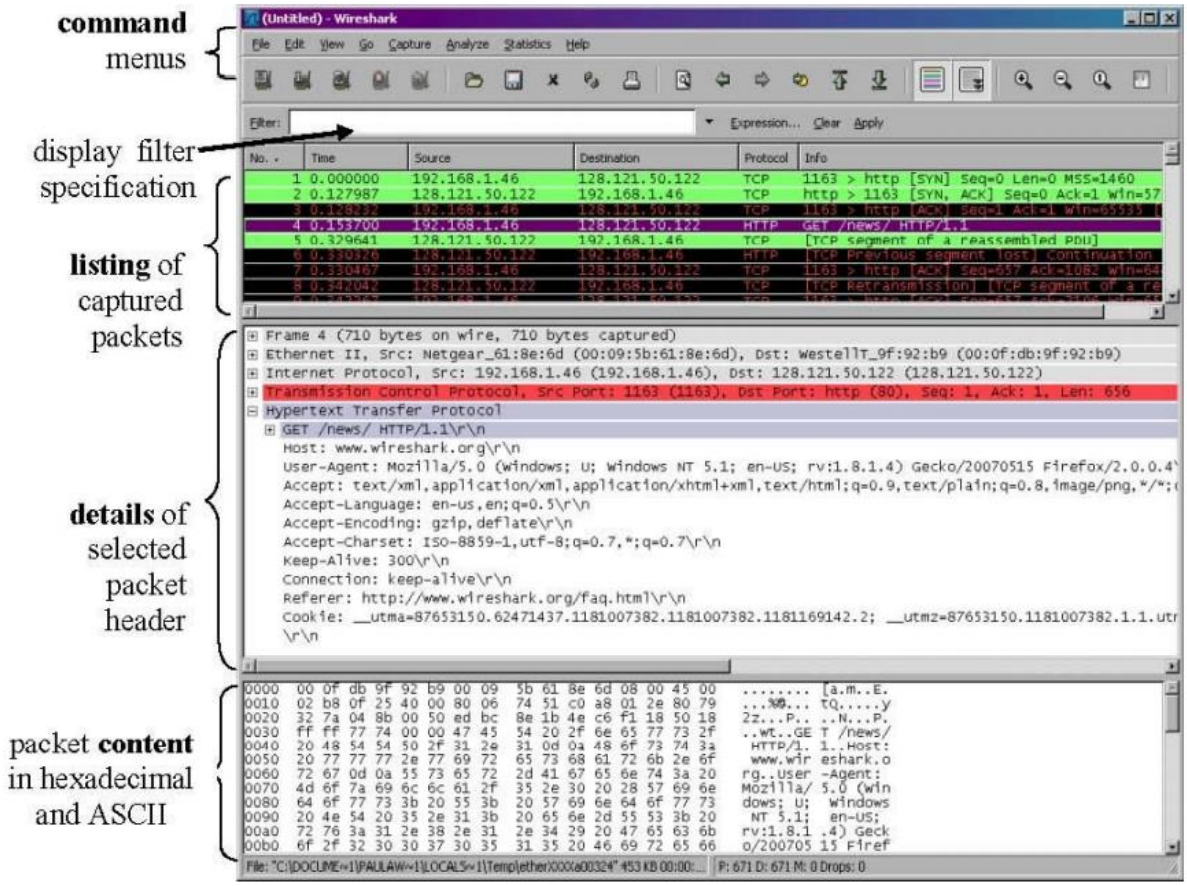
COMMUNICATION NETWORK LAB PROGRAMS

LAB 1

Packet Sniffing and Wireshark

The basic tool for observing the messages exchanged between executing protocol entities is called a packet sniffer. As the name suggests, a packet sniffer captures ("sniffs") messages being sent/received from/by your computer; it will also typically store and/or display the contents of the various protocol fields in these captured messages. A packet sniffer itself is passive. It observes messages being sent and received by applications and protocols running on your computer, but never sends packets itself. Similarly, received packets are never explicitly addressed to the packet sniffer. Instead, a packet sniffer receives a copy of packets that are sent/received from/by application and protocols executing on your machine.

Here we use a packet-sniffer called Wireshark. Wireshark (formerly known as ETHEREAL) is a free (open-source) packet sniffer/analyzer which is available for both UNIX-like (Unix, Linux, Mac OS X, BSD, and Solaris) and Windows operating systems. It captures packets from a network interface and displays them with detailed protocol information. Wireshark, however, is a passive analyzer. It only captures packets without manipulate them; it neither sends packets to the network nor does other active operations. Wireshark is not an intrusion-detection tool either. It does not give warning about any network intrusion. It, nevertheless, can help network administrators to figure out what is going on inside a network and to troubleshoot network problems. In addition of being an indispensable tool for network administrators, Wireshark is a valuable tool for protocol developers, who may use it to debug protocol implementations. It is also a great educational tool for computer-network students who can use it to see details of protocol operations in real time. Given below is the image of Wireshark Graphical User Interface, during packet capture and analysis.



Experiment

In this lab, we retrieve a web page and then, using Wireshark, capture packets.

- Start up your web browser and clear the browser's cache memory, but do not access any site yet.
- Open the Wireshark and start capturing. Use the filter box to capture only frames that the source or the sink protocol is HTTP.
- Note that you need to type http in lowercase in the filter box and click Apply. Now, go back to your browser, access one of your favorite web site.
- Stop capturing and save the captured file.

Questions

Using the first frame with the source protocol HTTP, answer the following questions.

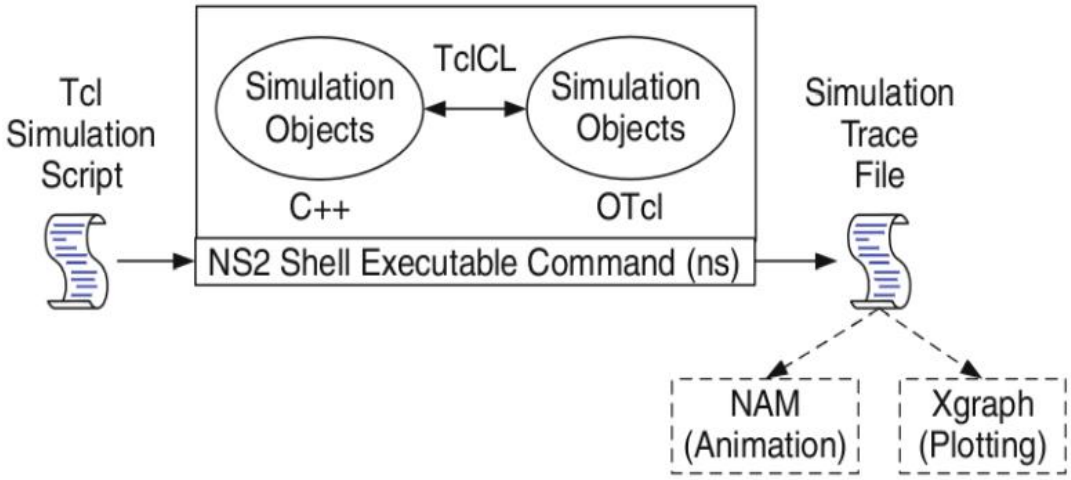
1. Is the frame an outgoing or an incoming frame?
2. What is the source IP address of the network-layer header in the frame?
3. What is the destination IP address of the network-layer header in the frame?
4. What is the total number of bytes in the whole frame?

5. What is the number of bytes in the Ethernet (data-link layer) header?
6. What is the number of bytes in the IP header?
7. What is the number of bytes in the TCP header?
8. What is the total bytes in the message (at the application layer)?

LAB 2
Introduction to NS2 and TCL scripting

Network Simulator (Version 2), widely known as NS2, is an event driven simulation tool that is useful in studying the dynamic nature of communication networks. NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviors. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2.

General Architecture of NS2



NS uses two languages as the simulator has two different kinds of things it needs to do. On one hand, detailed simulations of protocols require a system programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. For these tasks run-time speed is important and turn-around time (run simulation, find bug, fix bug, recompile, re-run) is less important. On the other hand, a large part of network research involves slightly varying parameters or configurations, or quickly exploring a number of scenarios. In these cases, iteration time (change the model and re-run) is more important. Since configuration runs once (at the beginning of the simulation), run-time of this part of the task is less important. ns meets both of these needs with two languages, C++ and OTcl.

TCL scripting

- Tcl is a general-purpose scripting language. [Interpreter]
- Tcl runs on most of the platforms such as Unix, Windows, and Mac.
- The strength of Tcl is its simplicity.
- It is not necessary to declare a data type for variable prior to the usage.

Wired TCL Script Components

1. Create the event scheduler
2. Open new files & turn on the tracing
3. Create the nodes
4. Setup the links
5. Configure the traffic type (e.g., TCP, UDP, etc)
6. Set the time of traffic generation (e.g., CBR, FTP)
7. Terminate the simulation

NS Simulator Preliminaries.

1. Initialization and termination aspects of the ns simulator.
2. Definition of network nodes, links, queues and topology.
3. Definition of agents and of applications.
4. The nam visualization tool.
5. Tracing and random variables.

Program

```
set ns [new Simulator]                  # Letter S is capital

set nf [open out.nam w]                 # open a nam trace file in write mode
set nt [open out.tr w]

$ns namtrace-all $nf                    # In TCL script files are called by the pointers initialized
$ns trace-all $nt                       # i.e., nf and nt respectively

set n0 [$ns node]                       # create nodes
set n1 [$ns node]

$ns duplex-link $n0 $n1 1Mb 8ms DropTail        # establishing links

set tcp [new Agent/TCP]                 # attaching transport layer protocols
$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]            # creating sink(destination) node
$ns attach-agent $n1 $sink
```

```
$ns connect $tcp $sink

set ftp [new Application/FTP]                # attaching application layer protocols
$ftp attach-agent $tcp

set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1

set sink1 [new Agent/TCPSink]
$ns attach-agent $n0 $sink1

$ns connect $tcp1 $sink1

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

proc finish {} {
        global ns nf nt
        $ns flush-trace                      # clears trace file contents
        close $nf
        close $nt
        exec nam out.nam &
        exit 0
        }

$ns at 0.1ms "$ftp start"
$ns at 0.5ms "$ftp1 start"
$ns at 2ms exit
$ns run
```

Steps for execution

Open gedit and type the program.

And program name when saved should have the extension " .tcl "
[root@localhost ~]# gedit lab1.tcl &

Run the simulation program
[root@localhost~]# ns lab1.tcl

Here "ns" indicates network simulator. The network animator window to visualize the simulation will open.

Now press the play button in the simulation window and the simulation will begins.

To see the trace file contents open the file as ,
[root@localhost~]# gedit lab1.tr or

[root@localhost~]# more lab1.tr

LAB 3
Simulate a three node point-to-point network with a duplex link between them for the following cases:

d. **One node a acts as the sender and receiver and the remaining nodes acts as a repeater**
e. **One node acts as the sender and two nodes acts receiver and data should flow in two different directions**
f. **Two nodes acts as sender and one node acts as receiver. Set queue size to 3 and vary the bandwidth (two cases) and count the packets being dropped**

NOTE: Use TCP as transport layer protocol and FTP as traffic source for all cases

Constraint: **In all the above cases, no node or link should be kept inactive at any instant of time**

a.

```
set ns [new Simulator]
$ns color 1 Red

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

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

$n0 color red
$n1 color blue

$ns duplex-link $n0 $n1 2Mb 4ms DropTail
$ns duplex-link $n1 $n2 2Mb 4ms DropTail

set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]
```

```
$ns attach-agent $n2 $sink

$ns connect $tcp $sink
$tcp set fid_ 1
$sink set fid_ 2

set ftp [new Application/FTP]
$ftp attach-agent $tcp

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

$ns at 0.1ms "$ftp start"
$ns at 4ms "finish"
$ns run
```

b.

```
set ns [new Simulator]
set nf [open sim.nam w]
$ns namtrace-all $nf

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

$ns duplex-link $n0 $n1 2Mb 4ms DropTail
$ns duplex-link $n1 $n2 2Mb 4ms DropTail

set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1
set tcp2 [new Agent/TCP]
$ns attach-agent $n1 $tcp2


set sink1 [new Agent/TCPSink]
$ns attach-agent $n0 $sink1
```

```
set sink2 [new Agent/TCPSink]
$ns attach-agent $n2 $sink2

$ns connect $tcp1 $sink1
$ns connect $tcp2 $sink2

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2

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

$ns at 0.1ms "$ftp1 start"
$ns at 0.1ms "$ftp2 start"
$ns at 4ms "finish"
$ns run
```

c.

```
set ns [new Simulator]
$ns color 1 Blue
$ns color 2 Red

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

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

$n1 color red
$n2 color red

$ns duplex-link $n1 $n0 10Mb 5ms DropTail
$ns duplex-link $n0 $n2 10Mb 5ms DropTail
$ns duplex-link-op $n0 $n1 orient right-up
```

```
$ns duplex-link-op $n0 $n2 orient left-down

$ns duplex-link-op $n0 $n1 queuePos 0.5
$ns duplex-link-op $n0 $n2 queuePos 0.5

$ns queue-limit $n1 $n0 3
$ns queue-limit $n2 $n0 3

set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1
set tcp2 [new Agent/TCP]
$ns attach-agent $n2 $tcp2

set sink1 [new Agent/TCPSink]
$ns attach-agent $n0 $sink1
set sink2 [new Agent/TCPSink]
$ns attach-agent $n0 $sink2

$ns connect $tcp1 $sink1
$ns connect $tcp2 $sink2

$tcp1 set fid_ 1
$tcp2 set fid_ 2

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2

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

$ns at 0.1ms "$ftp1 start"
$ns at 0.1ms "$ftp2 start"
$ns at 4ms "finish"
$ns run
```

LAB 4

Simulate a four-node point-to-point network and connect the link as follows: Apply TCP agent between n0 and n3. Apply relevant applications over TCP and UDP. Change bandwidth and latency (two cases) and determine the number of packets sent by two agents using awk script.

```
set ns [new Simulator]
$ns color 1 Blue
$ns color 2 Red

set nf [open sim.nam w]
set nt [open l4.tr w]
$ns namtrace-all $nf
$ns trace-all $nt

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

$n0 color red
$n1 color red
$n2 color brown

$ns duplex-link $n0 $n2 2Mb 8ms DropTail
$ns duplex-link $n1 $n2 1Mb 8ms DropTail
$ns duplex-link $n2 $n3 2Mb 8ms DropTail

$ns queue-limit $n0 $n2 3
$ns queue-limit $n1 $n2 3

set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink

$ns connect $tcp $sink

set udp [new Agent/UDP]
```

```
$ns attach-agent $n1 $udp

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

$ns connect $udp $null

$tcp set fid_ 1
$udp set fid_ 2

set ftp [new Application/FTP]
$ftp attach-agent $tcp
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp

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

$ns at 0.1ms "$ftp start"
$ns at 0.1ms "$cbr start"
$ns at 4ms "finish"
$ns run
```

.awk script

Open gedit and type awk program. Program name should have the extension ".awk "
[root@localhost ~]# gedit *filename*.awk &

After simulation is completed run awk file to see the output,
[root@localhost~]# awk –f *filename*.awk *filename*.tr

```
BEGIN {udp=0; tcp=0; drop=0}
{
if ($1=="r" && $5=="cbr")
{
```

```
udp++;
}
 else if ($1=="r" && $5=="tcp")
{
tcp++;
}
 else if ($1=="d" && $5=="tcp")
{
drop++
}
}
END {
printf("Number of packets sent by TCP = %d \n",tcp);
printf("Number of packets sent by UDP = %d\n",udp);
printf("Number of packets dropped by TCP = %d\n",drop);
}
```

LAB 5

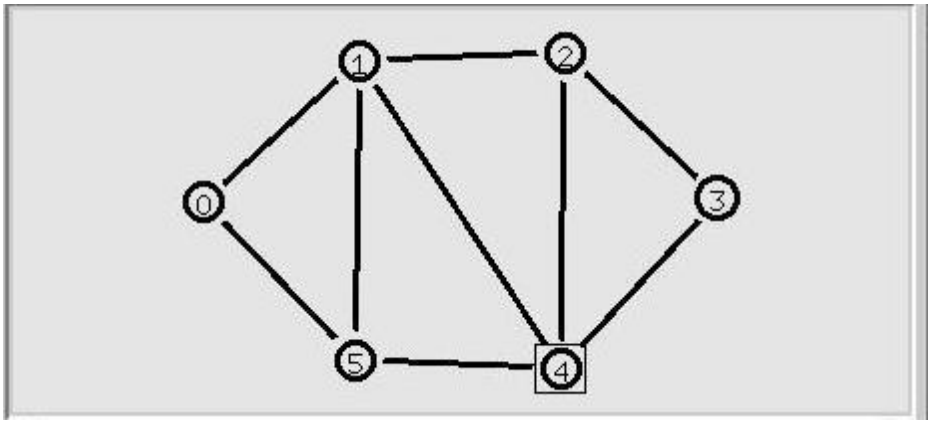
Assign *cost* between 0 & 1 as 5 and 5 & 4 as 3

Make node-3 as sink, node-1 and node-0 as UDP sources

a. **Simulate Distance Vector Routing protocol**
b. **Repeat the above for Link State Routing protocol awk script to count the number of packets**

- **make link 2-3 go down at 2.0 and link 1-4 at 2.8**
- **make link 1-4 up at 3.0 and 2-3 at 3.5**
- **make link 2-3 and 2-4 go down at 4.0**
- **stop simulation at 4.5 or 5**

Observe what happens at the beginning, when a link is down and up. Deduce relevant inference from your observation(s) and write appropriate conclusions



a.

```
set ns [new Simulator]
set nr [open thro.tr w]
$ns trace-all $nr
set nf [open thro.nam w]
$ns namtrace-all $nf

for { set i 0 } { $i < 6} { incr i 1 } {
set n($i) [$ns node]}
$ns color 1 Red
$ns color 2 Green


for {set i 0} {$i < 5} {incr i} {
$ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail }
$ns duplex-link $n(2) $n(4) 1Mb 10ms DropTail
$ns duplex-link $n(1) $n(5) 1Mb 10ms DropTail
$ns duplex-link $n(1) $n(4) 1Mb 10ms DropTail
$ns duplex-link $n(0) $n(5) 1Mb 10ms DropTail
$ns cost $n(0) $n(1) 5
$ns cost $n(5) $n(4) 3

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(3) $null0
```

```
$ns connect $udp0 $null0

set udp1 [new Agent/UDP]
$ns attach-agent $n(1) $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 200
$cbr1 set  interval_ 0.005
$cbr1 attach-agent $udp1
set null0 [new Agent/Null]
$ns attach-agent $n(3) $null0
$ns connect $udp1 $null0

$udp0 set fid_ 1
$udp1 set fid_ 2

$ns rtproto DV
# use LS instead of DV for Link State protocol
$ns rtmodel-at 2.0 down $n(2) $n(3)
$ns rtmodel-at 2.8 down $n(1) $n(4)
$ns rtmodel-at 3.0 up $n(1) $n(4)
$ns rtmodel-at 3.5 up $n(2) $n(3)
$ns rtmodel-at 4.0 down $n(2) $n(3)
$ns rtmodel-at 4.0 down $n(2) $n(4)

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

$ns at 1.0 "$cbr0 start"
$ns at 0.5 "$cbr1 start"
$ns at 5 "finish"
$ns run
```

b.

```
set ns [new Simulator]
set nr [open thro.tr w]
$ns trace-all $nr
```

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

$ns namtrace-all $nf
        proc finish { } {
        global ns nr nf
        $ns flush-trace
        close $nf
        close $nr
        exec nam thro.nam &
                exit 0
        }

for { set i 0 } { $i < 6} { incr i 1 } {
set n($i) [$ns node]}

for {set i 0} {$i < 5} {incr i} {
$ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail }
$ns duplex-link $n(2) $n(4) 1Mb 10ms DropTail
$ns duplex-link $n(1) $n(5) 1Mb 10ms DropTail
$ns duplex-link $n(1) $n(4) 1Mb 10ms DropTail
$ns duplex-link $n(0) $n(5) 1Mb 10ms DropTail

$ns cost $n(0) $n(1) 5
$ns cost $n(5) $n(4) 3

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(3) $null0
$ns connect $udp0 $null0

set udp1 [new Agent/UDP]
$ns attach-agent $n(1) $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 200
$cbr1 set  interval_ 0.005
$cbr1 attach-agent $udp1
set null0 [new Agent/Null]
```

```
$ns attach-agent $n(3) $null0
$ns connect $udp1 $null0

$ns rtproto LS
$ns rtmodel-at 2.0 down $n(2) $n(3)
$ns rtmodel-at 2.8 down $n(1) $n(4)
$ns rtmodel-at 3.0 up $n(1) $n(4)
$ns rtmodel-at 3.5 up $n(2) $n(3)
$ns rtmodel-at 4.0 down $n(2) $n(3)
$ns rtmodel-at 4.0 down $n(2) $n(4)

$udp0 set fid_ 1
$udp1 set fid_ 2
$ns color 1 Red
$ns color 2 Green

$ns at 1.0 "$cbr0 start"
$ns at 0.5 "$cbr1 start"
$ns at 5 "finish"
$ns run
```

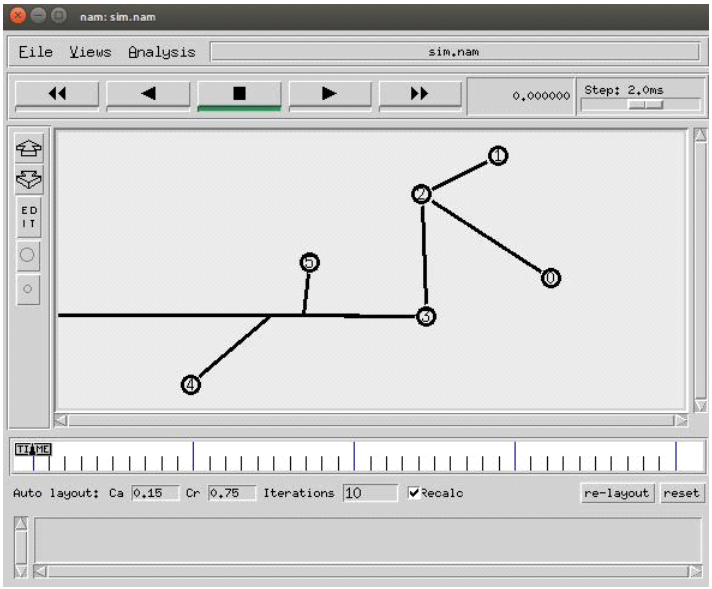
LAB 6

Simulate a 6 node ethernet LAN network. Change error rate and Data rate, compare the throughput. Make node-0 as TCP transmitter, node-1 as UDP transmitter and node-4 receiver & node-5 as NULL

- **Use ring topology: Connect each node to next node; node-5 to node-0**
- **Use bus topology for the tnework shown below**

  Use *set lan [$ns newLan "$n(3) $n(4) $n(5)" 2Mb 40ms LL Queue/DropTail MAC/802_3 Channel]* for bus connection

NOTE: Try for multiple cases of different data and error rates




```
set ns [new Simulator]
set nf [open lab6.nam w]
set nt [open lab6.tr w]
$ns namtrace-all $nf
$ns trace-all $nt

$ns color 1 Green
$ns color 2 Red

for { set i 0 } { $i < 6} { incr i 1 } {set n($i) [$ns node]}

$ns duplex-link $n(0) $n(2) 2Mb 4ms DropTail
$ns duplex-link $n(1) $n(2) 2Mb 4ms DropTail
$ns duplex-link $n(2) $n(3) 2Mb 4ms DropTail

set lan [$ns newLan "$n(3) $n(4) $n(5)" 1Mb 40ms LL Queue/DropTail MAC/802_3 Channel]

$ns duplex-link-op $n(0) $n(2) orient right-down
$ns duplex-link-op $n(1) $n(2) orient right-up
$ns duplex-link-op $n(2) $n(3) orient right

$ns queue-limit $n(2) $n(3) 20
$ns duplex-link-op $n(2) $n(3) queuePos 0.5
```

```
set err [new ErrorModel]
$err ranvar [new RandomVariable/Uniform]
$err drop-target [new Agent/Null]
$ns lossmodel $err $n(2) $n(3)

set tcp [new Agent/TCP]
$ns attach-agent $n(0) $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n(4) $sink
$ns connect $tcp $sink

set udp [new Agent/UDP]
$ns attach-agent $n(1) $udp
set null [new Agent/Null]
$ns attach-agent $n(5) $null
$ns connect $udp $null

set ftp [new Application/FTP]
$ftp attach-agent $tcp
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp

$tcp set fid_ 1
$udp set fid_ 2

proc finish {} {
        global ns nf nt
        $ns flush-trace
        close $nf
        close $nt
        exec nam sim.nam &
        set tcpsize [exec sh f1.sh]
        set tcpnum [exec sh f2.sh]
        set udpsize [exec sh f3.sh]
        set udpnum [exec sh f4.sh]
        set time_of_exec 124.00

        puts "Throughput of TCP is [expr $tcpsize * $tcpnum / $time_of_exec] bytes per sec \n"
        puts "Throughput of UDP is [expr $udpsize * $udpnum / $time_of_exec] bytes per sec
        \n"
        exit 0
```

```
        }

$ns at 0.0ms "$ftp start"
$ns at 0.0ms "$cbr start"
$ns at 124ms "finish"
$ns run
```

f1.sh

```
grep "^r" lab6.tr|grep tcp|tail -n 1|cut -d " " -f 6
```

f2.sh

```
grep "^r" lab6.tr |grep -c "tcp"
```

f3.sh

```
grep "^r" lab6.tr|grep "cbr"|tail -n 1| cut -d " " -f 6
```

f4.sh

```
grep "^r" lab6.tr |grep -c "cbr"
```

LAB 7
SIMULATION OF STOP AND WAIT PROTOCOL

```
set ns [new Simulator]
set nf [open stop.nam w]
$ns namtrace-all $nf
set f [open stop.tr w]
$ns trace-all $f

set n0 [$ns node]
set n1 [$ns node]
$ns at 0.0 "$n0 label Sender"
$ns at 0.0 "$n1 label Receiver"

$ns duplex-link $n0 $n1 0.2Mb 200ms DropTail
$ns duplex-link-op $n0 $n1 orient right
$ns queue-limit $n0 $n1 10

Agent/TCP set nam_tracevar_ true
set tcp [new Agent/TCP]
$tcp set window_ 1
$tcp set maxcwnd_ 1
$ns attach-agent $n0 $tcp
```

```
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $sink
$ns connect $tcp $sink

set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns add-agent-trace $tcp tcp
$ns monitor-agent-trace $tcp
$tcp tracevar cwnd_

$ns at 0.1 "$ftp start"
$ns at 3.0 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n1 $sink"
$ns at 3.5 "finish"

$ns at 0.0 "$ns trace-annotate \"Stop and Wait with normal operation\""
$ns at 0.05 "$ns trace-annotate \"FTP starts at 0.1\""
$ns at 0.11 "$ns trace-annotate \"Send Packet_0\""
$ns at 0.35 "$ns trace-annotate \"Receive Ack_0\""
$ns at 0.56 "$ns trace-annotate \"Send Packet_1\""
$ns at 0.79 "$ns trace-annotate \"Receive Ack_1\""
$ns at 0.99 "$ns trace-annotate \"Send Packet_2\""
$ns at 1.23 "$ns trace-annotate \"Receive Ack_2 \""
$ns at 1.43 "$ns trace-annotate \"Send Packet_3\""
$ns at 1.67 "$ns trace-annotate \"Receive Ack_3\""
$ns at 1.88 "$ns trace-annotate \"Send Packet_4\""
$ns at 2.11 "$ns trace-annotate \"Receive Ack_4\""
$ns at 2.32 "$ns trace-annotate \"Send Packet_5\""
$ns at 2.55 "$ns trace-annotate \"Receive Ack_5 \""
$ns at 2.75 "$ns trace-annotate \"Send Packet_6\""
$ns at 2.99 "$ns trace-annotate \"Receive Ack_6\""
$ns at 3.1 "$ns trace-annotate \"FTP stops\""

proc finish {} {
global ns nf
$ns flush-trace
close $nf
puts "running nam..."
exec nam stop.nam &
exit 0
}
$ns run
```

LAB 8

NETSIM EXPERIMENTS- I

a. Understand working of ARP, and IP forwarding within a LAN and across a router
b. Understand the working of "Connection Establishment" in TCP

LAB 9

NETSIM EXPERIMENTS- II

a. How many downloads can a WiFi access point simultaneously handle?
b. Plot the characteristic curve of throughput versus offered traffic for a Pure and Slotted ALOHA system
c. Study the working and routing table formation of Interior routing protocols, i.e. Routing Information Protocol (RIP) and Open Shortest Path First (OSPF)

LAB 10

NETSIM EXPERIMENTS- III

a. Study the hidden node problem in WLAN
b. Understand the working of basic networking commands (Ping, Route Add/Delete/Print, ACL)