

CS/ECE 757 – Computer Networks Fall 2004

Assignment 3: Programming with ns-2 Network Simulator

Instructions:

- Complete the homework in groups of two. Each group submits one solution, and obtains the same grade.
- The only help that you can use are your class notes and the textbook.
- Late submission penalty is 10% of the total grade per day.
- Be neat. Unreadable solutions (by our standards) will not be graded.
- Your submission must be typed.
- Please send questions to cs757@cs.virginia.edu

Note:

For this homework you need access to a UNIX computer account in the Computer Science Department. If you do not have a UNIX computer account in the Computer Science Department, you need to work with a partner who has an account.

Rationale: Unfortunately, most computers at UVA do not run the *ns* simulator. The installation of *ns* is time consuming and has to be customized for each installation. And in the homework assignment, you need to work on revised ns-2 software which is installed in /home/cs757/ns directory on the servers of the CS department. Therefore we recommend to finishing your homework on a CS departmental server.

Due Date: November 2, at the beginning of class.

Objective: In this homework, you will use the ns simulator for a comparison of congestion control algorithms of different versions of the TCP protocol. The relevant reading in the book “Computer Networking: A Top-Down Approach Featuring the Internet” for this homework is Section 3.7, and you can find additional background on TCP in Section 3.5. Detailed information can be found in:

[1] Van Jacobson, Congestion Avoidance and Control. In *Proceedings of ACM SIGCOMM '88*, pages 314-329, August 1988, Stanford, CA. Available at:

<http://www.acm.org/pubs/articles/proceedings/comm/52324/p314-jacobson/p314-jacobson.pdf>

[2] Jon Postel, Transmission Control Protocol. IETF RFC 0793, September 1981. Available at:

<http://www.ietf.org/rfc/rfc0793.txt>

Instructions

- Login to one of the CS UNIX departmental servers (mamba.cs.virginia.edu, cobra.cs.virginia.edu, or viper.cs.virginia.edu), and start an X-Windows session. Issue the following command to add ns to your “path”:
`export PATH=$PATH:/home/cs757/ns/ns:/home/cs757/ns/nam:/home/cs757/ns/xgraph/`
- When running a ns script, you may meet the following warning message:
Warning: no class variable Agent/TCPSink::bytes_
See tcl-object.tcl in tclcl for info about this warning.
It is not a problem and doesn't affect your results.

Problem 1: Write the following ns script and save it in a file `hw3-rfc793.tcl`

```
# Create a new simulator object
set ns [new Simulator]
# Enable the nam trace
$ns trace-all [open hw3.tr w]
$ns namtrace-all [open hw3.nam w]
# Create the following procedure which launches nam
proc finish {} {
    global ns
    $ns flush-trace
    puts "filtering..."
    #put below bold contents into one line.
    exec /home/cs757/ns/bin/tclsh8.3
    /home/cs757/ns/nam/bin/namfilter.tcl hw3.nam
    puts "running nam..."
    exec nam -a hw3.nam &
    exit 0
}
# Creates four nodes n0, n1, n2, n3
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

# Create the following topology, where
# n0 and n1 are connected by a duplex-link with
# capacity 5Mbps, propagation delay 20ms, dropping
# discipline "DropTail".
$ns duplex-link $n0 $n1 5Mb 20ms DropTail

# n1 and n2 are connected by a duplex-link with
# capacity 0.5Mbps, propagation delay 100ms, and
# dropping discipline "DropTail".
$ns duplex-link $n1 $n2 0.5Mb 100ms DropTail

# n2 and n3 connected by a duplex-link with capacity 5Mbps,
# propagation delay 20ms, dropping discipline "DropTail".
$ns duplex-link $n2 $n3 5Mb 20ms DropTail

# Create a bottleneck between n1 and n2, with a maximum queue
# size of 5 packets
$ns queue-limit $n1 $n2 5

# Instruct nam how to display the nodes
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n1 $n2 queuePos 0.5
```

```

# Establish a TCP connection between n0 and n3
set tcp [new Agent/TCP/RFC793edu]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink

# Create an FTP transfer (using the TCP agent)
# between n0 and n3
set ftp [new Application/FTP]
$ftp attach-agent $tcp

# Start the data transfer:
$ns at 0.1 "$ftp start"
$ns at 5.0 "$ftp stop"

# Launch the animation
$ns at 5.1 "finish"
$ns run

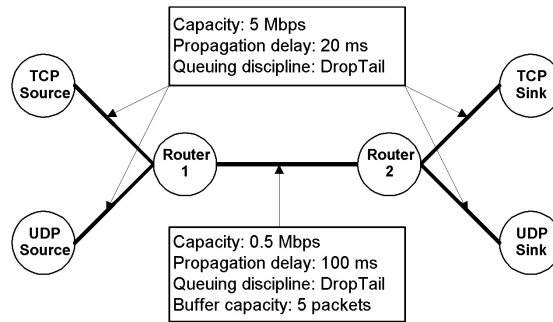
```

- a) Launch the simulator with your script by issuing the command
`ns hw3-rfc793.tcl`
- b) Observe what happens after the first packet is sent and acknowledged. Observe the packet losses that occur at router 1.
- c) In the script `hw3-rfc793.tcl`, replace the command containing
`set tcp [new Agent/TCP/RFC793edu]`
with:
`set tcp [new Agent/TCP]`
and save the modified script as `hw3-tahoe.tcl`
- d) Run the script `hw3-tahoe.tcl`. What are the two main differences to the results obtained with the script `hw3-rfc793.tcl`?

The two scripts examine some of the differences between the original TCP (from 1981), and a revision of TCP, called TCP-Tahoe (from 1988). Find two other variants of TCP developed from the 1990s, and explain how they work. (Note: There are at least *five* variants of the original TCP as described in RFC 793 that have been implemented.) What is the variant most commonly implemented in the major operating systems (i.e., Windows XP, FreeBSD, Linux, MacOS)?

Problem 2: Compatibility of TCP and UDP

1. Modify the script `hw3-tahoe.tcl` to simulate the following topology:



Both of the TCP and UDP sources should start transmitting at time 0.1 second, and stop at time 5.0 seconds.

Configure the UDP source to be a CBR source, sending packets of size 1000 bytes, at a rate of 0.1Mbps. Configure the UDP sink to be a LossMonitor agent. Keep the same TCP source and sink as in the script `hw3-tahoe.tcl`.

2. Use Chapter VIII of the ns tutorial (<http://www.isi.edu/nsnam/ns/tutorial/nsscript4.html#second>) to write a procedure that records the throughput of the TCP flow and the throughput of the UDP flow, starting a time 0.0 seconds, until time 5.0 seconds, with a sliding window of size 0.5 seconds. Using *xgraph*, plot the throughput obtained by both flows on the same graph. Submit your script and the plot. Make sure the axis and the curves are properly labeled.

Important Remark: We modified the simulator so that Agent/TCPSink has a “bytes_” variable that can be read by TCL scripts to determine the amount of data received, in a fashion similar to Agent/LossMonitor.

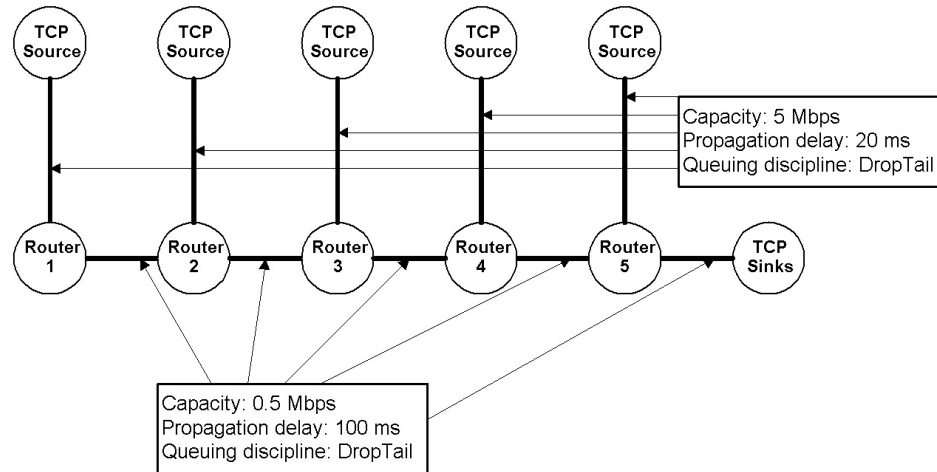
3. Modify the script and repeat the simulation so that the UDP source sends at a rate of

- 0.2 Mbps.
- 0.5Mbps, and
- 1 Mbps.

For each of these three simulations, plot the throughput obtained by both of the TCP and UDP flows on the same graph. Submit your plots and describe the results. Explain how TCP and UDP interact.

Problem 3: Fairness of TCP connections

1. Write a script simulating the following topology:



Remarks:

- In ns, you may attach multiple sinks to the same node.
- Do not limit the queue size at any of the links.

All sources start transmitting at time $t=0.1$ s, and stop transmitting at time $t=10$ s.

- Modify the procedure for recording the throughput you have used in part 2 of Problem 2 to monitor the throughput of all five TCP flows at the sinks, averaged over a sliding window of size 0.5 seconds. Plot the results on the same graph. Submit your script and the plots.
- Do all flows get a fair share of the available bandwidth? Provide a discussion of the fairness.