Networks — CSCI 125, Fall 2014
Instructors: Goodney
Lab 3
Due: 2:00PM on Thursday, October 9

Name: Bruce Yan (byan@hmc.edu)
DETER ID: hmc125ah, hmc125-ANGELA
Worked with: Angela Zhou

Problem 1 - Group Project: Fast, Reliable File Transfer:
`scp`. The 'secure copy program' `scp` is a standard tool on modern UNIX-like machines. It is used to copy files between machines, securely and reliably. However, as we will see, it does not always provide good throughput.

- We created an experiment in DETER with two computers together with a 100Mb/s link and 50ms initial delay. Below is the .ns file we used.

```
# lab3-part1.ns

# This is a simple ns script. Comments start with #.
set ns [new Simulator]
source tb_compat.tcl

# Define the 4 nodes in the topology
set nodeA [$ns node]
set nodeB [$ns node]

# Define the link and the LAN that connect the nodes
# This sets the link speed between nodeA and nodeB
# There is a 25ms round-trip delay
set link0 [$ns duplex-link $nodeB $nodeA 100Mb 50ms DropTail]

# Set the OS to Ubuntu1204-64-STD (modern)
tb-set-node-os $nodeA FBSD9-64-STD
tb-set-node-os $nodeB FBSD9-64-STD

# Enable routing (static) between all the nodes
$ns rtproto Static

# Instruct the simulator to start
# Go!
$ns run

# Make sure to check "Swap in Immediately"
```

- a
- b

- c

A copy of the NS file used to create this experiment can be found below:

```
# Tutorial.ns

# This is a simple ns script. Comments start with #.
set ns [new Simulator]
source tb_compat.tcl

# Define the 4 nodes in the topology
set nodeA [$ns node]
set nodeB [$ns node]

# Define the link and the LAN that connect the nodes
# This sets the link speed between nodeA and nodeB
# There is a 25ms round-trip delay
set link0 [$ns duplex-link $nodeB $nodeA 1Mb 25ms DropTail]

# Set the OS to Ubuntu1204-64-STD (modern)
tb-set-node-os $nodeA FBSD9-64-STD
tb-set-node-os $nodeB FBSD9-64-STD

# Enable routing (static) between all the nodes
$ns rtproto Static

# Instruct the simulator to start
# Go!
$ns run

# Make sure to check "Swap in Immediately"
```

Problem 2 - Gigabit links:

- I used `ping` to indeed verify the average RRT appears to be around 0.2 ms.
- Using TCP, the bandwidth appears to be around 938 Mbits/s. Using UDP, the speed appears to be 800 Mbits/s.
- The default window size is set to 64k. In most cases, the default window size will be fine, seeing as it's sufficiently small. However, in some cases, when the data that is being sent over is much less than the window size, the data won't be sent over by the transmission simply because the pipe is not yet full. This will cause delay in the transmission. Thus, a smaller window size is recommended.
- Using window size might be a good way to imitative delay, but I'm not sure if it will scale well. As mentioned previously, if the window size is too big, then the server will have to wait for the entire pipe to fill up before sending the data over. However, once the data is sent over, it gets sent over. There's no delay in that process. So having a variable window size might be good to simulate delay, but it doesn't appear to scale really well.
- When setting the TCP throughput to be very small (i.e. too small), it starts to affect the overall TCP throughput. According to my tests, when I set both the server and the client window size to be too small, the throughput reduced to nearly 400 Mbits/s.
- With gigabit links, the greater the delay, the more data we lose. It appears that a lot of bandwidth is not utilized efficiently. The lesser the delay, the better in terms of throughput performance and efficiency. I personally would not be happy with a 1GB DSL connection knowing I can only control the receive window size, especially if the server is on the other side of the country.