TCP Performance Measurements over GENI
CS 655: Introduction to Computer Networks
Fall 2015

Purpose:
The purpose of this experiment is to evaluate the performance of TCP over different network conditions using GENI.

Prerequisites:
Before beginning this experiment, you should be prepared with the following:

• You have GENI credentials to obtain GENI resources.
• You are able to reserve GENI resources.

• Download the attached RSpec file and save it on your machine. ([http://groups.geni.net/geni/attachment/wiki/UDTEExampleExperiment/udt.rspec?format=raw](http://groups.geni.net/geni/attachment/wiki/UDTEExampleExperiment/udt.rspec?format=raw)) Make sure to save in raw format.

• Create a new slice using the GENI Portal. Give your slice a name, e.g. GENITCP-YourLastName.

• After your slice is created, click the “Add Resources” button as shown below.

• In the “Choose RSpec” section as shown below, first check “File”, and click the “Choose File” button to upload the RSpec file that you just downloaded. Then click the “Reserve Resources” button at the bottom.
• After the resources are successfully reserved, you should be able to see a topology as below (at the bottom of the newly opened webpage), as well as details of these resources. 

**Note**: It may take several minutes before the resources are ready to log in.

![Topology Diagram](image)

**Install Iperf:**

Iperf is a tool to measure maximum TCP/UDP bandwidth (throughput), allowing the tuning of various parameters and characteristics. Iperf reports bandwidth, delay jitter, datagram loss. Follow the following steps to install Iperf:

• Log into node PC1 and node PC2 in separate windows.

• Install Iperf on PC1 and PC2 using the following command:

  ```bash
  sudo apt-get install iperf
  ```

**Setting the TCP version:**

First we will change the TCP congestion-control algorithm to TCP Reno:

• To check the TCP version running on PC1 and PC2, type following in the command line window for PC1 and PC2:

  ```bash
  cat /proc/sys/net/ipv4/tcp_congestion_control
  ```

• You should see the TCP version running on your VM. If it is not ‘reno’, type following to change it to Reno:

  ```bash
  sudo sysctl net.ipv4.tcp_congestion_control=reno
  ```
• Also make sure that you are using Selective Acknowledgement (sack) for TCP. To check if you are using sack, type following on PC1 and PC2:

```bash
cat /proc/sys/net/ipv4/tcp_sack
```

If you get output ‘1’, that means TCP sack is enabled. If you get output ‘0’, that means TCP sack is not enabled. To enable TCP sack, type:

```bash
sudo sysctl net.ipv4.tcp_sack=1
```

**Running Iperf:**

We will use PC1 as server and PC2 as client.

• Run `iperf` server on PC1 using the follow command:

```bash
iperf -s
```

You should see:

```
wyf@pc1:$ iperf -s
```

```
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
```

• Run `iperf` client on PC2 to connect to PC1:

```bash
iperf -c pc1
```

You should see an output on the console that looks like this:

```
wyf@pc2:$ iperf -c pc1
```

```
Client connecting to pc1, TCP port 5001
TCP window size: 16.0 KByte (default)
```

```
[ 3] local 192.168.2.2 port 38121 connected with 192.168.2.1 port 5001
[ ID] Interval     Transfer      Bandwidth
[ 3]  0.0-10.0 sec  1.10 GBytes  942 Mbits/sec
```

We can see that 942 Mbits/sec is the measured bandwidth (throughput). We are interested in this value.

**Adjusting link characteristics:**

In this experiment, you'll be changing the characteristics of the link and measuring how they affect TCP performance.
• Log into your 'delay' node as you do with any other node. Then, on your delay node, use this command:

    sudo ipfw pipe show

You'll get something like this:

    -bash-2.05b$ sudo ipfw pipe show
    60111: 1000.000 Mbit/s  1 ms  50 sl. 1 queues (1 buckets) droptail
       mask: 0x00 0x00000000/0x0000 -> 0x00000000/0x0000
    BKT Prot ___Source IP/port____ ___Dest. IP/port____ Tot_pkt/bytes Pkt/Byte Drp
       0 ip  207.168.149.200/0  195.120.134.8/6  814909 53764016 0 0 0
    60121: 1000.000 Mbit/s  1 ms  50 sl. 1 queues (1 buckets) droptail
       mask: 0x00 0x00000000/0x0000 -> 0x00000000/0x0000
    BKT Prot ___Source IP/port____ ___Dest. IP/port____ Tot_pkt/bytes Pkt/Byte Drp
       0 ip  207.168.143.172/0  195.83.8/8/6  1627500 2463702990 0 0 0

This information shows the internal configuration of the "pipes" used to emulate network characteristics. (Your output may look different, depending on the version of ipfw installed on your delay node. In any case, the information you need is on the first line of output for each pipe.)

You'll want to make note of the two pipe numbers, one for each direction of traffic along your link. In the example above, they are 60111 and 60121.

There are three link characteristics we'll manipulate in this experiment: bandwidth (capacity), delay, and packet loss rate. You'll find their values listed in the ipfw output above. The link bandwidth appears on the first line immediately after the pipe number. It's 1000Mbps in the example shown above. The next value shown is the delay, 1 ms in the example above. The packet loss rate (PLR) is omitted if it's zero, as shown above. If non-zero, you'll see something like `plr 0.000100` immediately after the "50 sl." on the first output line.

It is possible to adjust the parameters of the two directions of your link separately, to emulate asymmetric links. In this experiment, however, we are looking at symmetric links, so we'll always change the settings on both pipes together.

• Set your link parameters to use **maximum** bandwidth, 5ms delay and 0.01% packet loss (0.0001):

    sudo ipfw pipe 60111 config bw 0 delay 5 plr 0.0001
    sudo ipfw pipe 60121 config bw 0 delay 5 plr 0.0001
Throughput:

Theoretically, the throughput of TCP connection can be approximated by:

\[
\text{Throughput} = \frac{1.22 \text{ MSS}}{\text{RTT} \sqrt{\text{plr}}}
\]

where the Maximum Segment Size (MSS) = 1460 Bytes on a link with MTU of 1500 Bytes, and the **RTT value can be calculated using the Ping application.** This analytical model captures loss detection by duplicate acknowledgments but not due to retransmission timeouts (see lecture notes).

We will use the iPerf application to measure the TCP throughput for different link conditions, and compare the values of measured throughput with the analytical throughput values calculated using the equation above.

What to hand in:

- Measure the values of throughput for different values of packet loss. **Fill the table below using the measured values as well as the calculated values.**

(Values in parenthesis are calculated values of throughput you get from the equation above.)

<table>
<thead>
<tr>
<th>Delay: 5 ms</th>
<th>plr= 0.0001</th>
<th>plr=0.001</th>
<th>plr=0.01</th>
</tr>
</thead>
<tbody>
<tr>
<td>_______ Mb/s</td>
<td>_______ Mb/s</td>
<td>_______ Mb/s</td>
<td></td>
</tr>
<tr>
<td>( _______ Mb/s)</td>
<td>( _______ Mb/s)</td>
<td>( _______ Mb/s)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Delay: 10 ms</th>
<th>plr= 0.0001</th>
<th>plr=0.001</th>
<th>plr=0.01</th>
</tr>
</thead>
<tbody>
<tr>
<td>_______ Mb/s</td>
<td>_______ Mb/s</td>
<td>_______ Mb/s</td>
<td></td>
</tr>
<tr>
<td>( _______ Mb/s)</td>
<td>( _______ Mb/s)</td>
<td>( _______ Mb/s)</td>
<td></td>
</tr>
</tbody>
</table>

- Comment on how close your measured throughput values are to their analytical counterparts.
- Try at least two higher packet loss values (e.g., 2\% and 5\%) and comment on the validity of the above analytical model.