ITS413: Internet Technologies and Applications

Assignment 1 Final Report

By:
Penpisut Poowanaviroch
(5222780710)

Chitchanok Jantarat
(5222780934)

Nitchan Sombatpaiwan
(5222782138)

March 28, 2012

By submitting this report all members of the group listed above agree that each member has contributed approximately equal amounts to designing and performing experiments, as well as to preparing this report. All members agree that this report accurately reflects the experiments conducted by the group members, and is their own work (not works of other groups).

Sirindhorn International Institute of Technology
Thammasat University

Contents

Chapter 1

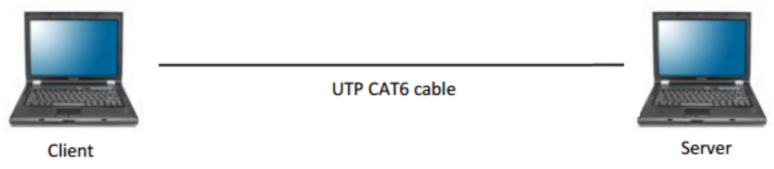
Aims

1. To find out which factors have impact on TCP performance.
2. To study the concept of flow control, error control and congestion control on TCP.
3. To understand the bandwidth delay product and TCP.
4. To prove the fairness of TCP.
5. To study how to use linux command line interface.

Chapter 2

Network Diagram

All experiments are conducted using the following network topology. Two laptops are connected directy through a UTP CAT6 LAN cable at 1000 Mbps data rate.



Chapter 3

Equipment Specification

Server

SONY VAIO VPCEA16FH
CPU : Intel Core i5-520M 2.40 GHz
Memory : 4 GB DDR3
HDD : Hitachi HTS54505 500GB 5400RPM SATA3
WLAN Interface : Intel Centrino Advanced-N 6200 (IEEE 802.11 b/g/n)
LAN Interface : Marvell Yukon Optima 88E8059 (10/100/1000 Mbps)
OS: Ubuntu Linux 11.04 x64

Client

Sony VAIO VPCEG18FH
CPU : Intel Core i5-2410M 2.30 GHz
Memory : 8 GB DDR3
HDD : OCZ-AGILITY3 120GB SSD SATA3
WLAN Interface : Atheros AR9285 (IEEE 802.11 b/g/n)
LAN Interface : Atheros AR8151 (10/100/1000 Mbps)
OS: Linux Mint 12 x64

Detailed Network Interfaces Information

Server

Wired Ethernet Interface: Marvell Yukon Optima 88E8059

```
04:00.0 Ethernet controller: Marvell Technology Group Ltd.
Yukon Optima 88E8059 [PCIe Gigabit Ethernet Controller with AVB] (rev11)
        Subsystem: Sony Corporation Device 9071
        Flags: bus master, fast devsel, latency 0, IRQ 41
        Memory at f2220000 (64-bit, non-prefetchable) [size=16K]
        I/O ports at a000 [size=256]
        Expansion ROM at f2200000 [disabled] [size=128K]
        Capabilities: [48] Power Management version 3
        Capabilities: [5c] MSI: Enable+ Count=1/1 Maskable- 64bit+
        Capabilities: [c0] Express Legacy Endpoint, MSI 00
        Capabilities: [100] Advanced Error Reporting
        Capabilities: [130] Device Serial Number 34-db-08-ff-ff-49-42-54
        Kernel driver in use: sky2
        Kernel modules: sky2
```

Client

Wired Ethernet Interface: Atheros AR8151

```
09:00.0 Ethernet controller: Atheros Communications AR8151 v2.0
Gigabit Ethernet (rev c0)
        Subsystem: Sony Corporation Device 908a
        Flags: bus master, fast devsel, latency 0, IRQ 43
        Memory at d4100000 (64-bit, non-prefetchable) [size=256K]
        I/O ports at 2000 [size=128]
        Capabilities: [40] Power Management version 3
        Capabilities: [48] MSI: Enable+ Count=1/1 Maskable- 64bit+
        Capabilities: [58] Express Endpoint, MSI 00
        Capabilities: [6c] Vital Product Data
        Capabilities: [100] Advanced Error Reporting
        Capabilities: [180] Device Serial Number ff-9a-92-39-78-84-3c-ff
        Kernel driver in use: atl1c
        Kernel modules: atl1c
```

Chapter 4

Experiments and Results

There are three sets of experiment conducted in this assignment.

- Impact of packet drops on TCP performance
- TCP and the bandwidth delay product
- Multiple TCP sessions

4.1 Impact of packet drops on TCP performance

4.1.1 Methodology

Let client IP address is 10.10.10.100 and server IP address is 10.10.10.10
Assume that both computers have iperf installed.

1. Start one instance of iperf server in TCP mode using default parameter on server computer.

   ```
   $ iperf -s
   ```

2. On client computer,do the experiment by starting an iperf client in TCP mode and specify the test time(duration) to be 1 minute.

   ```
   $ iperf -c 10.10.10.10 -t 60
   ```

   Run this command three times and note down the results.

3. On server computer, try to use iptables to drop the incoming packet.

   ```
   $ sudo iptables -A INPUT -m statistic --mode random --probability 0.01 -j DROP
   ```

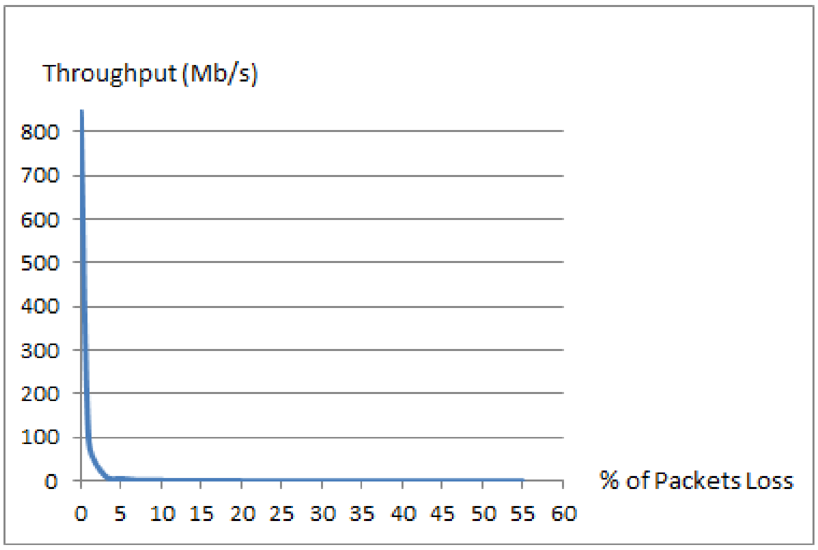
   This command means randomly drop 1% of incoming packet.

4. Perform step 2 again.
5. Clear all iptables rules using following command

   ```
   $ sudo iptables --flush
   ```

6. Repeat step 3 again using 0.03 as probability parameter.
7. Perform step 2 again.
8. Try changing probability parameter and repeat the experiment until all of these values are used.
   0.03, 0.07, 0.09, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50, 0.55

4.1.2 Result and Discussion



From the figure above, when there is no packet loss, TCP can get maximum throughput (about 850 Mb/s). If one percent of packet loss occurred, the throughput dramatically dropped to under 100 Mb/s, which is approximately 10% of maximum throughput. When the packet loss rate increased to be five percents, the throughput is around 6 Mb/s and will go less than a megabits per second if packet loss is 15%.

The reasons why maximum throughput cannot reach 1000 Mb/s are due to the overhead in the protocol stack such as TCP congestion control algorithm that will softly increase the sending rate (not fully use network capacity immediately when start) and some limitation of computer hardware.

For packet loss issue, if there are packet loss arose, TCP congestion control will try to decrease the sending rate since it thought that there is congestion developed in the network. Moreover, TCP has error control mechanism which responsible for retransmitting all packets that are not arrrived successfully to destination.

4.1.3 Conclusion

As described above, packet loss has high impact on TCP performance since TCP has error control mechanism for retransmitting lost data and congestion control that will slow down data sending rate when packet loss is detected. All of these make the time for sending whole data larger. Hence, longer time used means lower performance of the network.

4.2 TCP and the bandwidth delay product

4.2.1 Methodology

Let client IP address is 10.10.10.100 and server IP address is 10.10.10.10
Assume that both computers have iperf installed.

1. On both client(sender) and server(receiver), disable offloading so TCP/IP will not shifted its functionality to the network interface card.

   ```
   $ sudo ethtool -K eth0 gso off
   $ sudo ethtool -K eth0 gro off
   $ sudo ethtool -K eth0 tso off
   ```

   This will disable tcp-segmentation-offload, generic-segmentation-offload and generic-receive-offload. To ensure that they are disabled correctly, use the following command to view the configuration values.

   ```
   $ sudo ethtool -k eth0
   ```

   The value of tcp-segmentation-offload, generic-segmentation-offload and generic-receive-offload should be off.

2. On client(sender) set the TCP congestion control algorithm to *Reno*

   ```
   $ sudo sysctl net.ipv4.tcp_congestion_control=reno
   ```

3. On server(receiver) turn off automatic tuning of the TCP receive buffer size

   ```
   $ sudo sysctl net.ipv.tcp_moderate_rcvbuf=0
   ```

4. On both client(sender) and server(receiver) use *tc* to set the delay (RTT) on the link

   ```
   $ sudo tc qdisc add dev eth0 root netem delay 0.5ms
   ```

5. Measure the TCP throughput of different receive buffer size

   On client(sender), use:

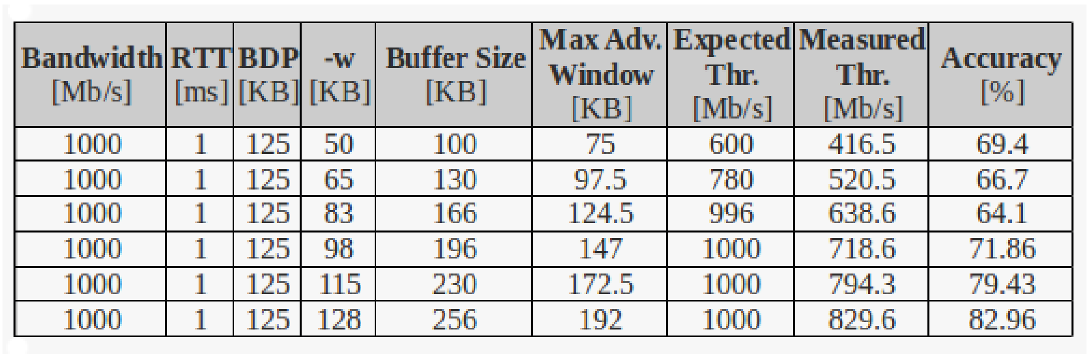
   ```
   $ ipert -c 10.10.10.10 -t 30
   ```

   On server(receiver), use:

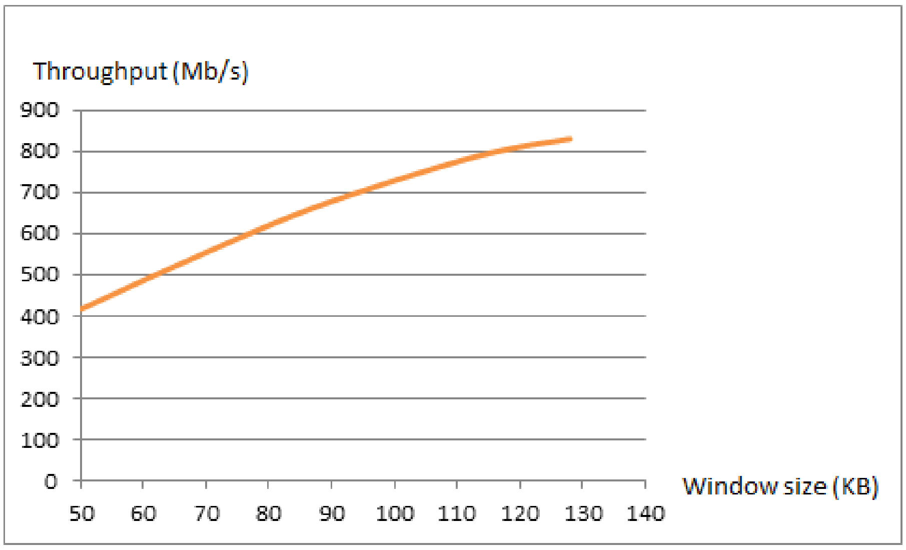
   ```
   $ iperf -s -w 50K
   ```

   Repeat window size (-w) with 65, 83, 98, 115 and 128K. Then, start the iperf client again for each window size value.

4.2.2 Result and Discussion



Bandwidth [Mb/s]	**RTT** [ms]	**BDP** [KB]	**-w** [KB]	**Buffer Size** [KB]	**Max Adv. Window** [KB]	**Expected Thr.** [Mb/s]	**Measured Thr.** [Mb/s]	**Accuracy** [%]
1000	1	125	50	100	75	600	416.5	69.4
1000	1	125	65	130	97.5	780	520.5	66.7
1000	1	125	83	166	124.5	996	638.6	64.1
1000	1	125	98	196	147	1000	718.6	71.86
1000	1	125	115	230	172.5	1000	794.3	79.43
1000	1	125	128	256	192	1000	829.6	82.96



- From the table above, 3 variables are fixed
  Bandwidth = 1000 Mb/s
  RTT = 1ms (using 0.5ms delay on client and server)
  *BDP = Bandwidth * RTT = 125 Kbytes*

- Window size (-w) is set to 50,65,83,98,115,128 respectively
- Buffer size is doubled from window size because the kernel wants to make sure there is enough space for storing received segments and information about the TCP connection.
- For maximum advertised window, the kernel allocates 75% from whole buffer size.
- Expected Throughput comes from
  *Expected Throughput = Maximum Advertised Window * Bandwidth*
- For the 4th, 5th and 6th row, expected throughput is equal to 1000 Mb/s since it is limited by the bandwidth size (Gigabit Ethernet).
- If the maximum advertised window is equal to or greater than BDP, then client (sender) can get maximum throughput. If less than, the throughput is maximum advertised window divided by RTT.
- Accuracy or performance is percent of measured(real) throughput in comparison to expected throughput.
  *Accuracy = (Measured Throughput / Expected Throughput) * 100*

4.2.3 Conclusion

From this experiment, we can prove that if the maximum advertised window is equal to or greater than BDP, maximum throughput can be achieved as sender does not have to wait for an ACK before send more data packet. If the maximum advertised window is less than BDP, the throughput will be *Maximum advertised window / RTT.*
The reason is that sender have to wait for an ACK back before it can send another data packet so the real throughput will be less than the former one.

4.3 Multiple TCP sessions

4.3.1 Methodology

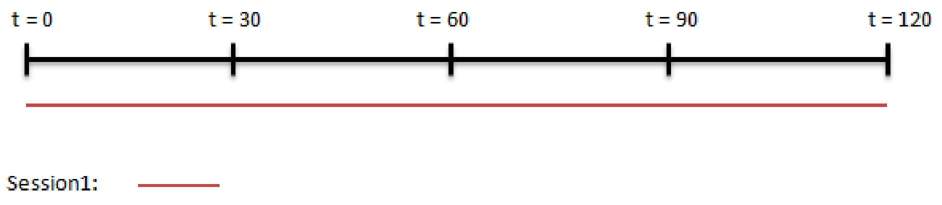
Let client IP address is 10.10.10.100 and server IP address is 10.10.10.10
Assume that both computers have iperf installed.

1. On server computer, start one instance of iperf server in TCP mode using default parameter.

   ```
   $ iperf -s
   ```

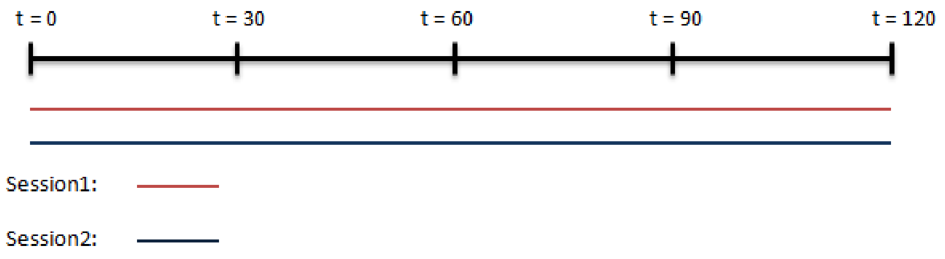
2. On client computer, create shell script files that will start multiple instances of iperf TCP client at vary time duration.

   a) One iperf instance that start immediately and run for 120 seconds.

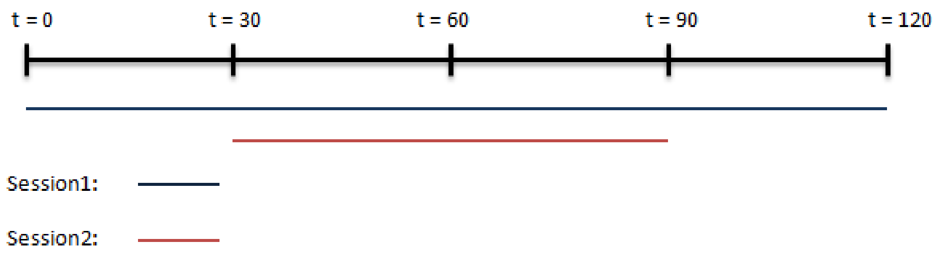
   ```
   gnome-terminal --tab -e "iperf -c 10.10.10.10 -t 120"
   ```

   b) Two iperf instances which start immediately at the same time and run for 120 seconds.

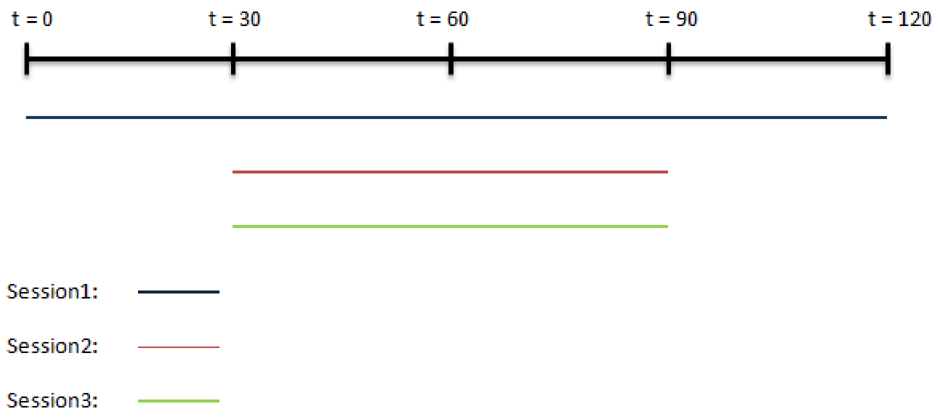
   ```
   gnome-terminal --tab -e "iperf -c 10.10.10.10 -t 120"
   gnome-terminal --tab -e "iperf -c 10.10.10.10 -t 120"
   ```

c) Two iperf instances, the first one will start immediately and run for 120 seconds. Another one will start 30 seconds after the initialization of the first one and runs for 60 seconds.



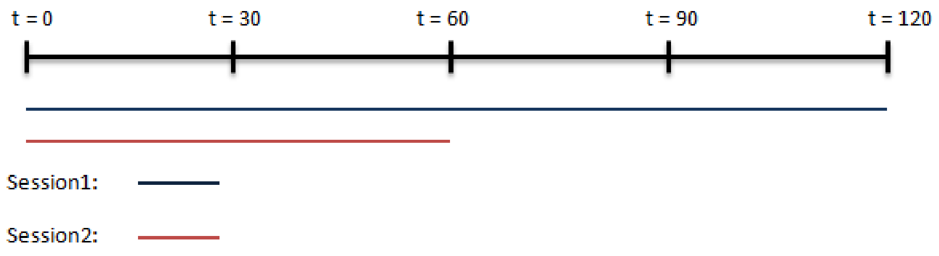
```
gnome-terminal --tab -e "iperf -c 10.10.10.10 -t 120"
sleep 30s
gnome-terminal --tab -e "iperf -c 10.10.10.10 -t 60"
```

d) Three iperf instances, the first one will start immediately and run for 120 seconds. Other two instances will concurrently start 30 seconds after the beginning of the first one and runs for 60 seconds.



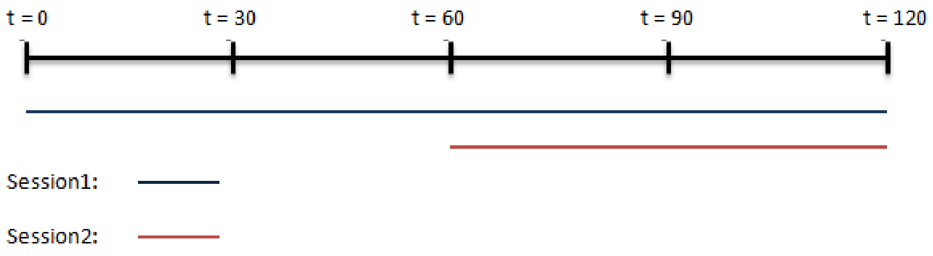
```
gnome-terminal --tab -e "iperf -c 10.10.10.10 -t 120"
sleep 30s
gnome-terminal --tab -e "iperf -c 10.10.10.10 -t 60"
gnome-terminal --tab -e "iperf -c 10.10.10.10 -t 60"
```

e) Two iperf instances, start immediately at the same time. The first one runs for 120 seconds and the second one runs for 60 seconds.



```
gnome-terminal --tab -e "iperf -c 10.10.10.10 -t 120"
gnome-terminal --tab -e "iperf -c 10.10.10.10 -t 60"
```

f) Two iperf instances, the second instance will start 60 seconds later. The duration is 120 seconds and 60 seconds respectively.



```
gnome-terminal --tab -e "iperf -c 10.10.10.10 -t 120"
sleep 60s
gnome-terminal --tab -e "iperf -c 10.10.10.10 -t 60"
```

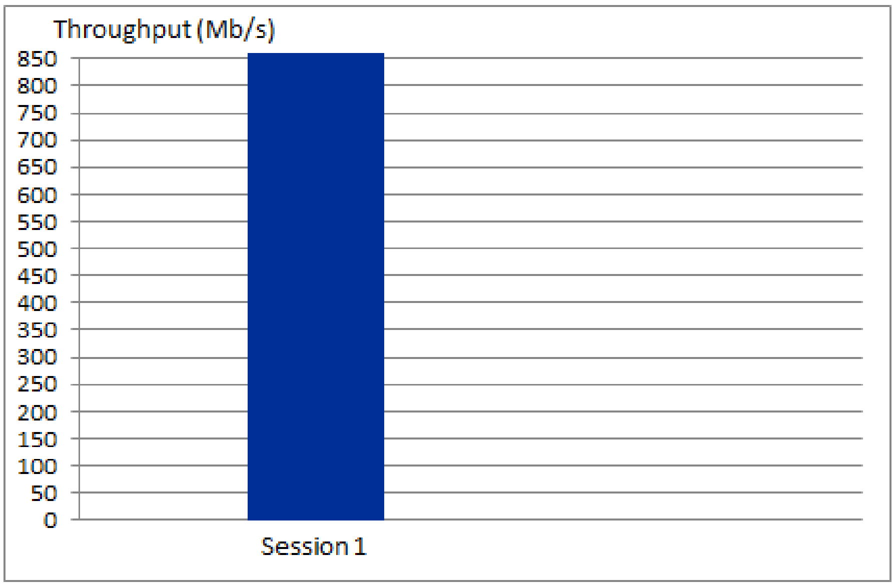
3. Save each file with *.sh* extension.

4. Execute each file consecutively (wait until it finish before executing another file).

```
$ bash filename.sh
```

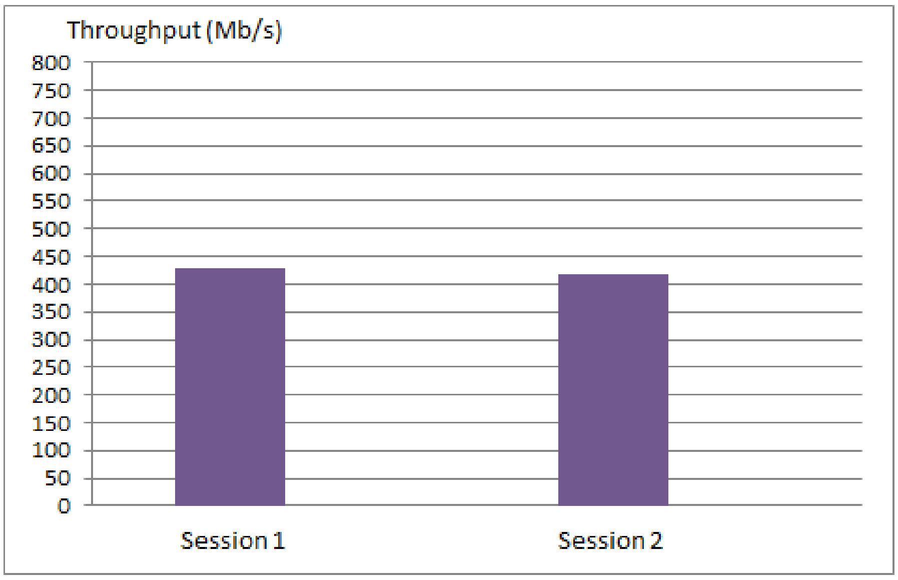
Where "filename" is the shell script file name.

4.3.2 Result and Discussion



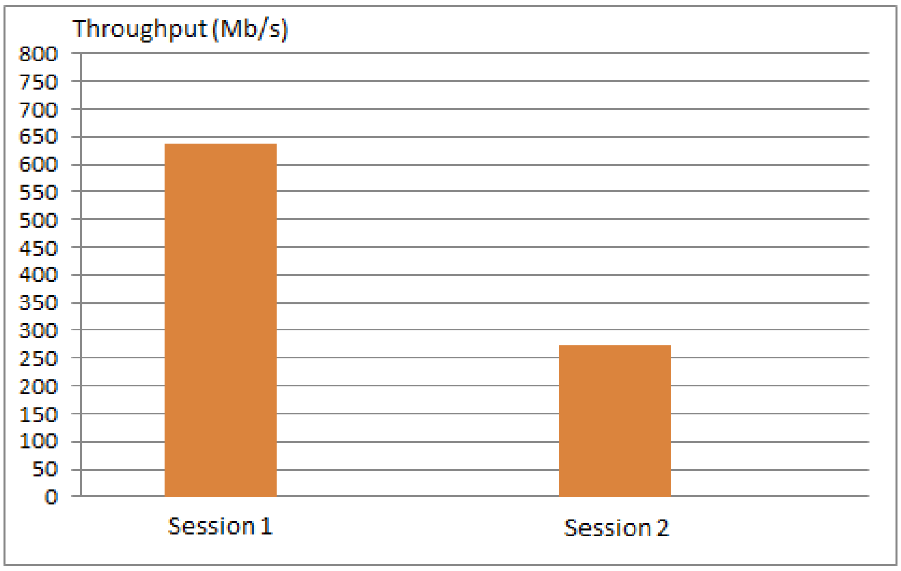
a) One iperf instance that runs for 120 seconds.

In this experiment, only one TCP session is using the network. Therefore, it can utilize all network capacity (about 850 Mb/s) which means it can send as fast as it want but not exceed the window size limit.



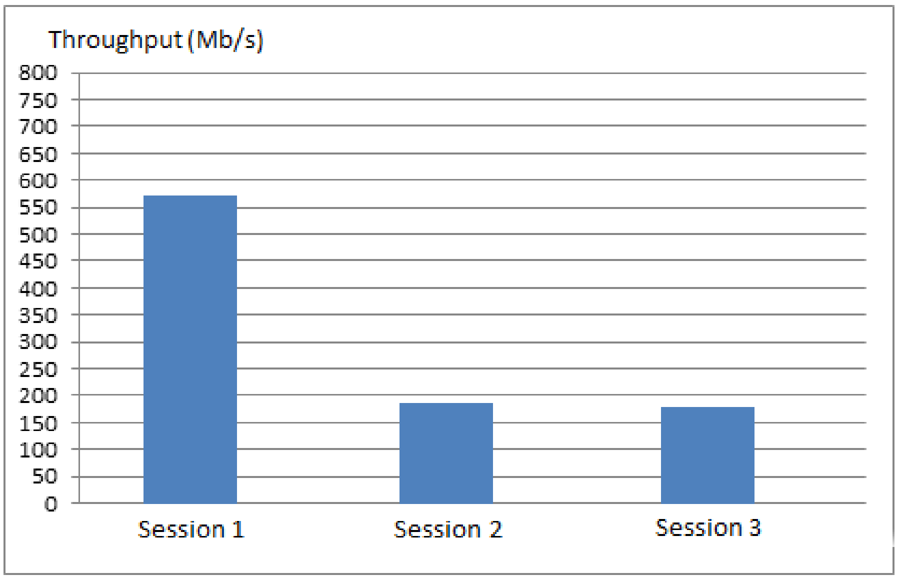
b) Two iperf instances which start at the same time and run for 120 seconds.

As shown in the figure above, TCP has fairness among sessions. In this case, both session start sending at the same time and congestion control algorithm of each session continuously increases the sending rate. When the sending rate is too high, packet loss will occur. Congestion control of the session that has packet loss knows there are congestion take place in the network so it has to decrease the sending rate. From the result, the average throughput of two sessions is not much different. This show that packet loss is almost equal and the decrement of sending rate of a session is close to each other.



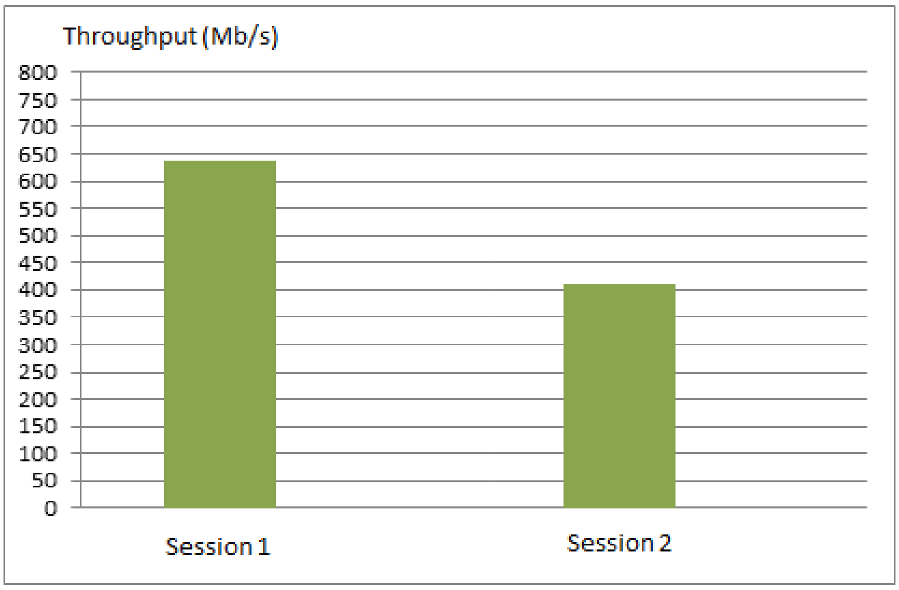
c) Two iperf instances, the first one will start immediately and run for 120 seconds. Another one will start 30 seconds later and runs for 60 seconds.

First session start sending data before the second session. Sending rate is increasing uninterruptedly until packet loss occur. The second session start sending 30 seconds later so the sending rate is behind the first one. Moreover, the second one comes into the network when the network usage is quite high so it suffered with high packet loss rate. This makes its congestion control to decrease the sending rate even if the rate is not so high. In addition, the first one has longer duration of using network so it can increase the sending rate more. Consequently, the second session gets lower throughput than the first one.



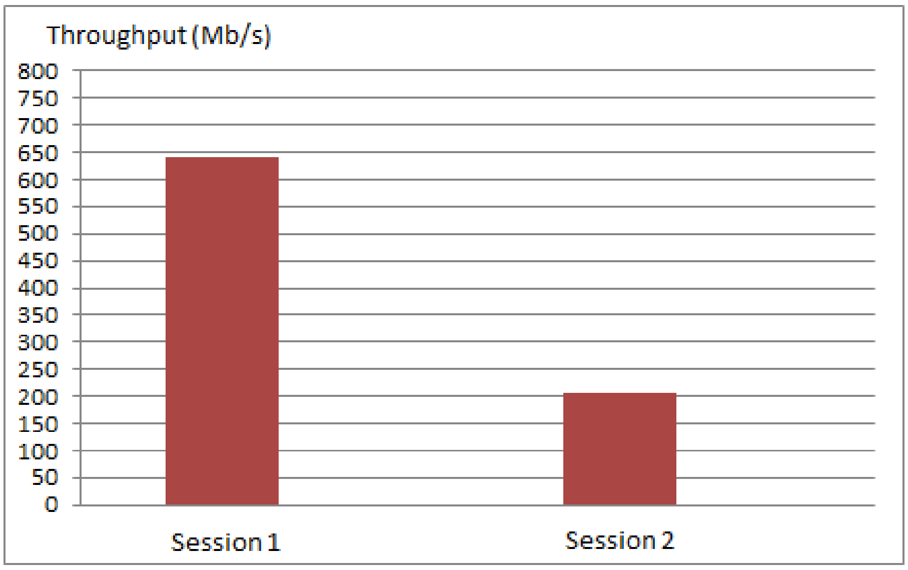
d) Three iperf instances, the first one will start immediately and run for 120 seconds. Other two instances will start 30 seconds later and runs for 60 seconds.

The last two sessions get similar throughput which are much lower than the first session. If we look at Session2 and Session3 as one session, it it like one long session with one short session in between. The explanation in the former case c) can be used; and if we consider only Session2 and Session3, this is similar to the b) case. Two session with the same duration start at the same time get almost equal throughput.



e) Two iperf instances, start at the same time. The first one runs for 120 seconds and the second one runs for 60 seconds.

In this scenario, Session2 gets lower throughput but not as much as in the case c) because both session start at the same time when the network utilization is very low. Their congestion control continuously increase the sending rate in parallel. When packet loss event happened with any session, its congestion control will try to slow down the sending rate to avoid congestion. This may similar to the case b) but the running time of Session2 is shorter than Session1. Thus, the first session can use high sending rate when Session2 ended which leads to higher thoughput of whole session.



f) Two iperf instances, the second one will start 60 seconds later. The duration is 120 seconds and 60 seconds respectively.

For this example, the second session starts at 60 seconds after the first one. However, both sessions end at the same time. The result obviously shows that Session2 get much lower throughput by virtue of its start time is at the state that network usage is very high, resulted in high packet loss rate which cause congestion control to decrease sending rate more often. Lower throughput indicates higher packet loss.

4.3.3 Conclusion

TCP has fairness among session is some scenario. For example, if all session start at the same time and has same duration, average throughput of each session will naerly the same. But if each session start at different time or has different duration, fairness may not be achieve as longer sessions or earlier started sessions can get higher throughput. When packet loss occurred in any session, its congestion control thought that there are congestion in the network. Therefore, the sending rate of that section will be decreased. Furthermore, TCP has error control mechanism to retransmit lost data packets. These are the reasons why throughput falls down when there are many session concurrently using the network. In conclusion, number of sessions significantly impacts on TCP performance.

Chapter 5

References

- Bandwidth delay product experiment setup procedures from sandilands website.
  http://sandilands.info/sgordon/impact-of-bandwidth-delay-product-on-tcp-throughput
- Instructions for using iptables to drop packets from sandilands website.
  http://sandilands.info/sgordon/dropping-packets-in-ubuntu-linux-using-tc-and-iptables
- Advices and suggestion from Dr.Steven Gordon.
- Original shell script for running multiple iperf sessions by Punsanga K. of Group 05.