Report on Assignment 2

TCP Performance

Prepared for

Asst. Prof. Dr. Steve Gordon

Prepared by

Kaelkrittaya Trurktham ID 5122780612
Wasamas Leakpech ID 5122781131
Bulakorn Phansuwan ID 5122791650

ITS413 Internet Applications

School of Information, Computer, and Communication Technology

Sirindhorn International Institute of Technology

Thammasat University

Semester 2, Academic Year 2010

OBJECTIVE

To study network parameters having an impacts on TCP performance

METHODOLOGY

Equipment

1. Server computer
    - Hardware specification: Intel Core2Duo T6600 2.20GHz CPU; 4GB RAM; Marvell Yukon 88E8040T PCI-E Fast Ethernet NIC
    - Operating system: Ubuntu 10.10 with Linux kernel 2.6.35
2. Client computer
    - Hardware specification: Intel Core2Duo T5750 2.00GHz CPU; 3GB RAM; Realtek RTL810IE PCI-E Fast Ethernet NIC
    - Operating system: Ubuntu 10.10 with Linux kernel 2.6.35
3. Cross-over Ethernet cable
4. Software
    - Installed software: *iperf*
    - Linux built-in software: *ifconfig*, *tc*; *iptables*

Experimental Setup

Throughout the assignment, the server computer was directly connected to the client computer using the cross-over Ethernet cable as shown in *Figure 1*. By using *ifconfig*, the server computer's Ethernet interface (eth0) was set to own the IP address 10.10.10.1, where the client computer's Ethernet interface (eth0) was set to own the IP address 10.10.10.2. By using *iperf*, the server computer was set to listen on the TCP port 50123 and the client computer was connected to that port on the dynamically-allocated-by-kernel TCP port.

iperf, *tc*, and *iptables* – providing different network facilities—were used to configure network parameters that have an impact on TCP performance. The selection of the software and their commands to use will also be mentioned separately in each task.

The bandwidth reported by *iperf* running at the server computer was observed and interpreted as the network throughput—an indicator of TCP performance.

Figure 1: Ethernet LAN topology

Task 1: Single TCP session with varied application/protocol parameters

1.1) Window Size

Experimental Procedure

1. Configure the server computer to listen with varied[##] TCP window size.
   On the terminal, run: *iperf -s -p 50123 –w ##*
2. Configure the client computer to run for 30 seconds with the TCP window size of 10 KB.
   On the terminal, run: *iperf -c 10.10.10.1 -p 50123 -t 30 –w 10000*
3. Observe the throughput of the network.
4. Configure the server computer to listen with the length of data of 10 KB.
   On the terminal, run: *iperf -s -p 50123 –w 10000*
5. Configure the client computer to run for 30 seconds with varied[##] TCP window size.
   On the terminal, run: *iperf -c 10.10.10.1 -p 50123 -t 30 –w ##*
6. Observe the throughput of the network.

Experimental Result

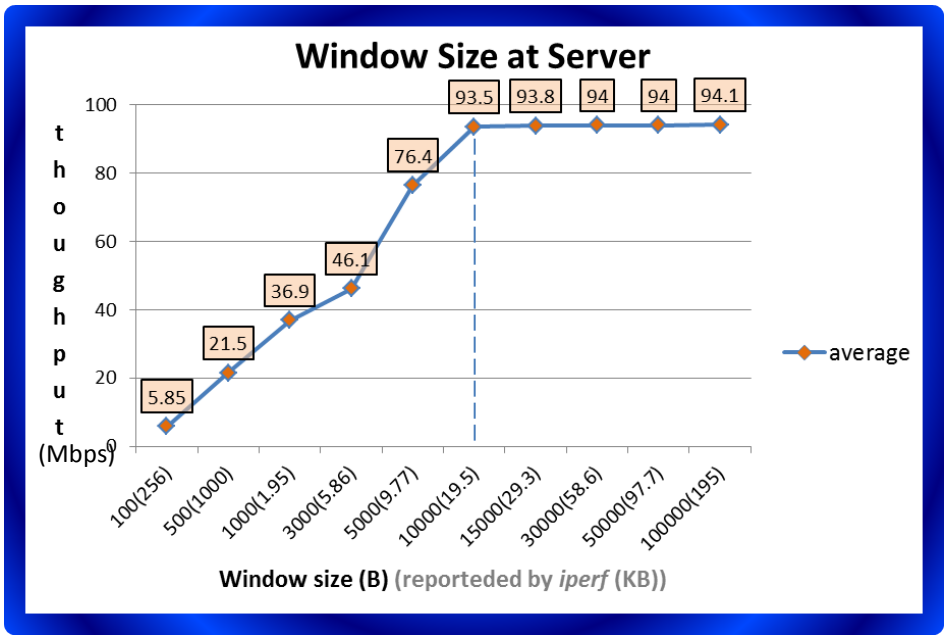

Figure 2: the throughput of the network when the receiver's TCP window size was varied

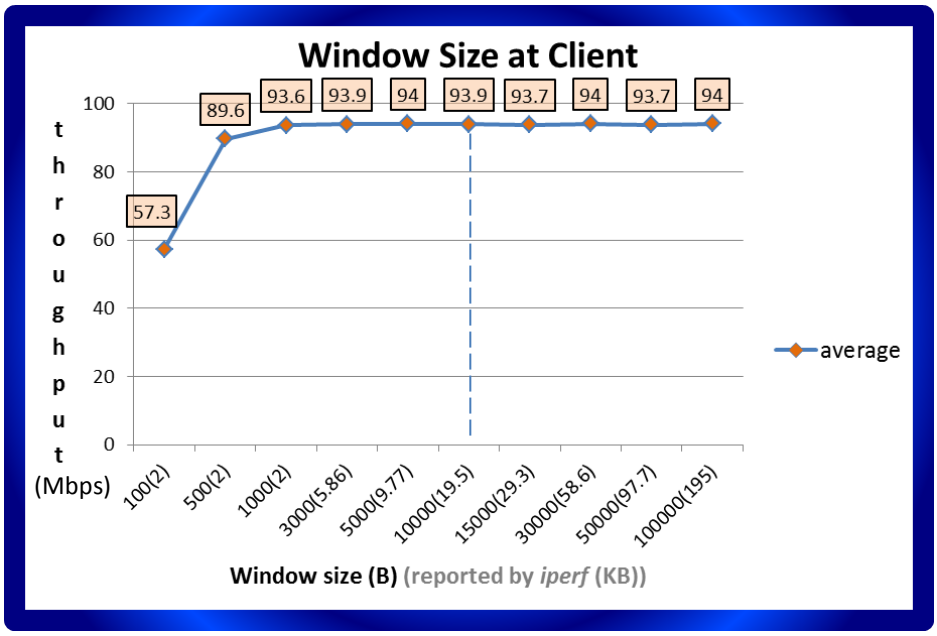

Figure 3: the throughput of the network when the sender's TCP window size was varied

Discussion

According to Figure 2, while the size of window at the client computer (the sender) was fixed at 10 KB, the throughput increased as the window size at the server computer (the receiver) increased. Until the window reached the same size as those of the client computer, 10 KB, the throughput went up to 94 Mbps and stayed constant at this rate although the window size still increased. This shows that the maximum capacity of the network was 94 Mbps and can be reached when the window size at the receiver was equal or higher than the window size at the sender.

To compare, while the size of window at the server computer (the receiver) was fixed at 10 KB according to Figure 3, the throughput increased as the window size at the client computer (the sender) increased but with slower rate than the case shown in Figure z1. Until the window reached the same size as those of the client computer, 10 KB, the throughput went up to the maximum of 94 Mbps and stayed constant at this rate although the window size still increased. This means that the maximum throughput can be reached when the window size at the sender was equal or higher than the window size at the receiver.

To conclude, the TCP window size of both the sender and the receiver contribute to network throughput. If the receiver's window size is smaller than the sender's, the surplus capacity at the sender is useless since the receiver can only hold that small capacity and does not allow the sender to overflow the TCP buffer (flow control mechanism). If the receiver's window size is equal or larger than the sender's, the surplus capacity at the receiver is also useless since the sender can only generate that small capacity for the receiver.

1.2) Length of Data

Experimental Procedure

1. Configure the server computer to listen with varied[##] length of data.
   On the terminal, run: *iperf -s -p 50123 –l ##*
2. Configure the client computer to run for 30 seconds with the length of data of 64 KB.
   On the terminal, run: *iperf -c 10.10.10.1 -p 50123 -t 30 –l 64*
3. Observe the throughput of the network.
4. Configure the server computer to listen with the length of data of 64 KB.
   On the terminal, run: *iperf -s -p 50123 –l 64*
5. Configure the client computer to run for 30 seconds with varied[##] length of data.
   On the terminal, run: *iperf -c 10.10.10.1 -p 50123 -t 30 –l ##*
6. Observe the throughput of the network.

Experimental Result

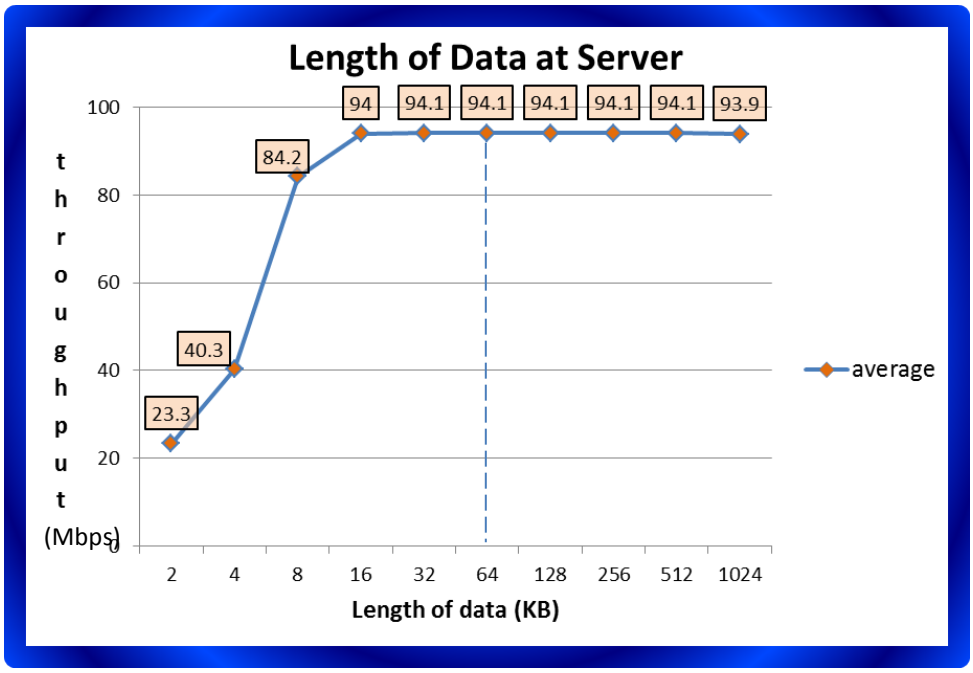

Figure 4: the throughput of the network when the receiver's length of data was varied

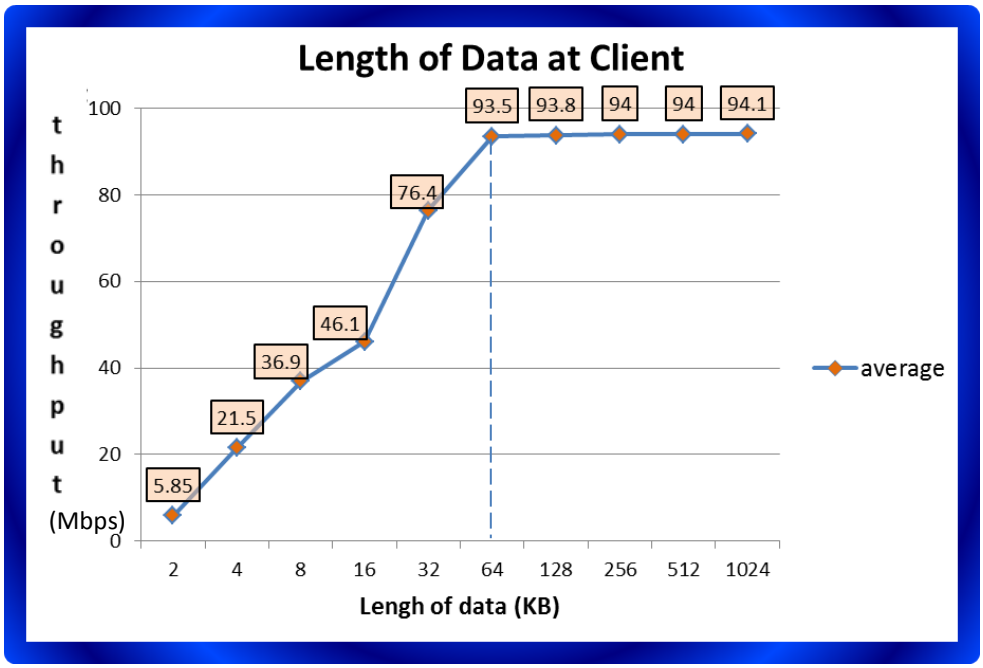

Figure 5: the throughput of the network when the sender's length of data was varied

Discussion

This experiment was divided into 2 cases. For the first case, the receiver fixed the window size to 97.7 KB along with the length of the data to 64 KB, while the receiver varied the length of the data. According to Figure 4, the throughput of the network tended to exponentially increase as the receiver's length of the data increased. The throughput was stable after it reached its maximum at the point where length of the data equal to 64 KB which is very close to the window size of about 85 KB. In this case, maximum throughput of the network was 94 Mbps and can be obtained when the length of the data at the receiver was approximately equal to or greater than the window size of the sender.

For the second case, the sender used default window size of 85.3 KB and fixed the length of the data to 64 KB, while the sender varied the length of the data this time. According to Figure 5, the throughput of the network looked very similar to Figure 4. That is the throughput reached the maximum and was stable the point where length of the data equal to 64 KB which is very close to the window size of about 85 KB.

To conclude, length of the data, which is the size of the data either the sender writes to TCP buffer at a time or the receiver reads from TCP buffer at a time, affects TCP throughput. Bigger length of the data means more data can be written or read by the sender or the receiver, respectively. As a result, the throughput increases as this length increases. The big length of the data, if bigger than the TCP window size, will increase no more throughput because there are not enough data in the buffer to be read or not enough space in the buffer to be written such a large size of data at a time.

Task 2: Single TCP session with varied network/link parameters

2.1) Link Data Rate

Experimental Procedure

1. Configure the server computer to listen with default configurations.
   On the terminal, run: *iperf -s -p 50123*

2. Configure the client computer to run for 30 seconds with varied[##] the data rate.
   On the terminal, run: *sudo tc qdisc add dev eth0 root tbf rate ##kbit latency 50ms burst 5kb*
   On the terminal, run: *iperf -c 10.10.10.1 -p 50123 -t 30*

3. Observe the throughput of the network.

Experimental Result

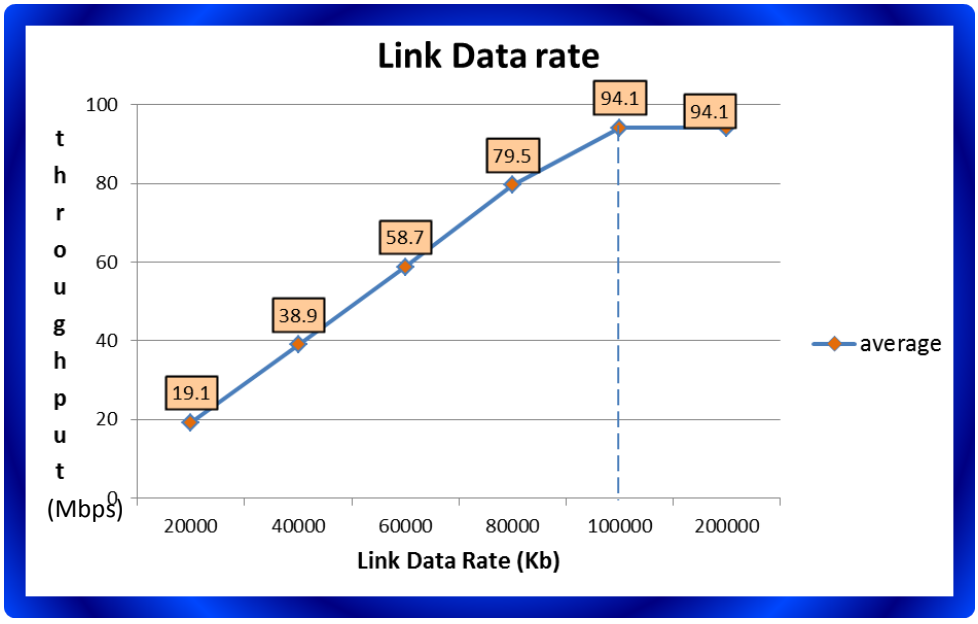

Figure 6: the throughput of the network when various link data rates

Discussion

According to Figure 6, the observed throughput increased as the configured link data rate increased with very less smaller value. The trend was as mentioned until the link data rate reached 100 Mbps. Higher configured link data rate than 100 Mbps did not make the throughput increase further but stayed at the maximum value of 94 Mbps.

To conclude, the high throughput of the network can be reached by setting high link data rate. The actual achieved throughput, however, is a little less than the link data rate due to packet transmission overhead such as packet headers, which results in less actual data able to be sent. The throughput is also restricted by the hardware specification of host computers, Ethernet cards, and Ethernet cables to some maximum valid link data rate.

2.2) Link Delay

Experimental Procedure

1. Configure the server computer to listen with default configurations.
   On the terminal, run: *iperf -s -p 50123*

2. Configure the client computer to run for 30 seconds with varied## delay.
   On the terminal, run: *sudo tc qdisc add dev eth0 root tbf netem delay ##ms 10ms*
   On the terminal, run: *iperf -c 10.10.10.1 -p 50123 -t 30*

3. Observe the throughput of the network.

Experimental Result

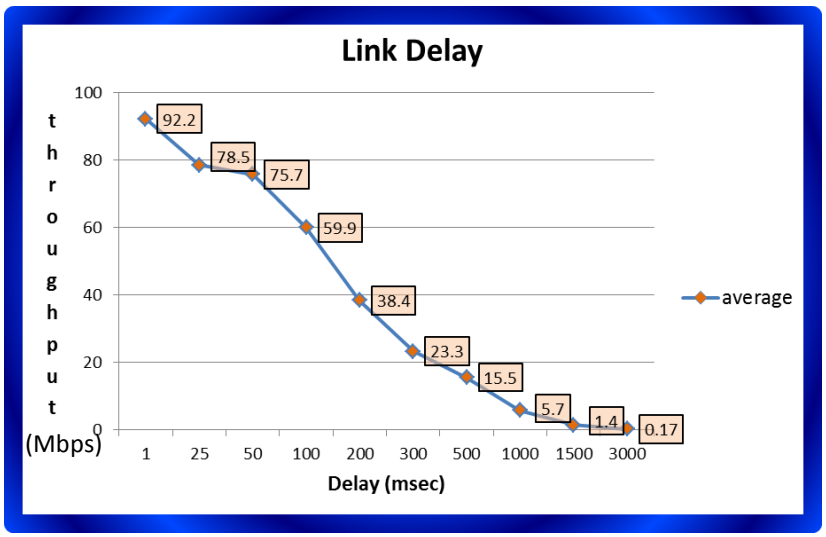

Figure 7: the throughput of the network when various link delays

Discussion

According to Figure 7, the throughput of the network reached its maximum value of 94 Mbps when the link had no delay. When delay was present in the link, the throughput linearly decreased as the link delay increased.

To conclude, the link delay such as the delay at routers in the Internet causes the throughput to reduce. This is because the delay causes the round trip time of each packet to increase. If that increase in the round trip time exceeds the packet timeout specified by TCP, meaning that the receiver does not receive the packet early enough to send a corresponding ACK back so that the sender is notified of successful transmission, the receiver considers this situation as packet loss, and as a result does retransmission of that packet. It is the retransmission itself that reduces the throughput of the network. When the link delay increases, many more packets are considered lost and many more unnecessary retransmissions badly spoil the overall throughput.

2.3) Packet Dropped

Experimental Procedure

1. Configure the server computer to listen with default configurations
   but varied## the number/probability of packets to be dropped.
   On the terminal, run: *sudo iptables -A INPUT -m statistic --mode random --probability ## -j DROP*
   On the terminal, run: *iperf -s -p 50123*

2. Configure the client computer to run for 30 seconds.
   On the terminal, run: *iperf -c 10.10.10.1 -p 50123 -t 30*

3. Observe the throughput of the network.

4. Repeat Step 2 and 3 but change the client computer to run for 60 seconds.
   On the terminal, run: *iperf -c 10.10.10.1 -p 50123 -t 60*

Experimental Result

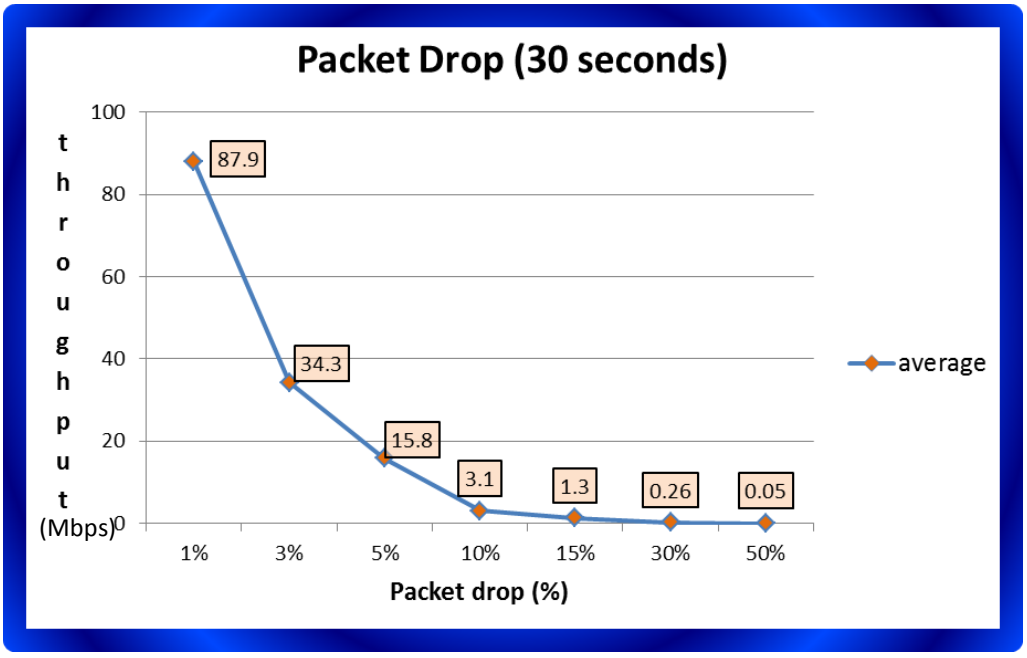

Figure 8: the throughput of the network when some packets were dropped during the test duration of 30 seconds

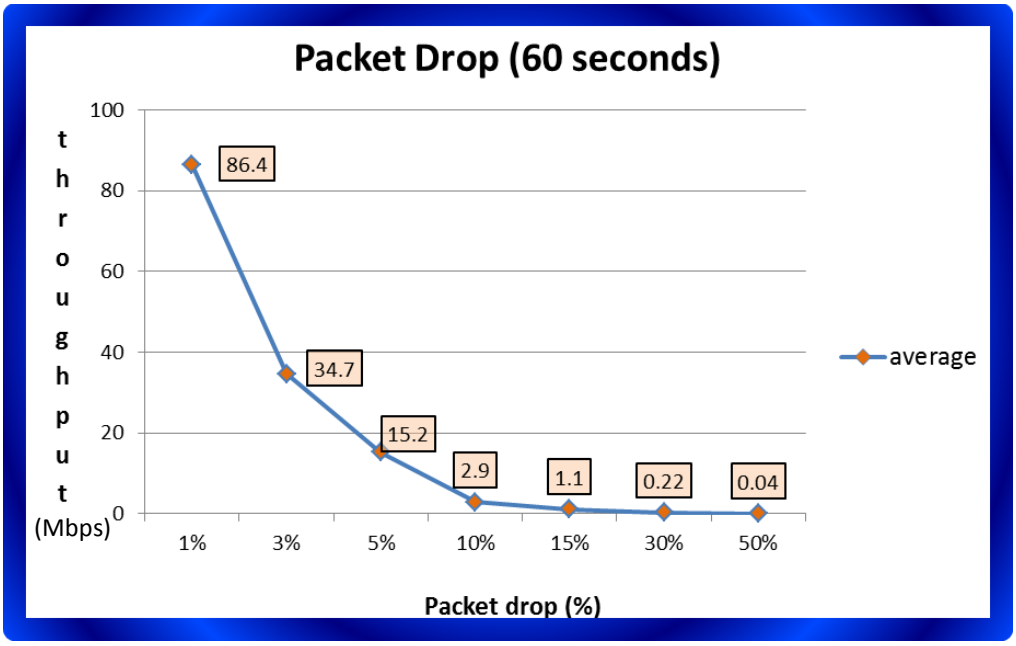

Figure 9: the throughput of the network when some packets were dropped during the test duration of 60 seconds

Discussion

According to Figure 8, the throughput of the network reached its maximum value of 94 Mbps when there were no packets dropped. When there were some packets dropped in the network, the throughput exponentially decreased as the dropping rate increased.

Similarly, when there were some packets dropped but the test duration was increased from 30 to 60 seconds, the throughput also exponentially decreased as the dropping rate increased.

To conclude, the packets dropped such as by routers in the Internet when the router's buffer is full causes the throughput to reduce. The reason is that the packets dropped never reach the receiver. If the receiver does not know about the packets to received, it ignores until the sender waiting for corresponding ACKs times out and retransmits those packets. Another case is that the dropped packets are expected by the receiver—the receiver knows that there should be specific packets coming to fulfill the order of received packets. If this happens, the receiver continuously sends ACKs for retransmission of missing packets to the sender. For both cases, the retransmission causes the throughput to reduce and the duration of the test did not affect the reduction of the throughput since the probability is used to drop packets, meaning that the number of packets dropped would increase with respect to the actual received packets.However, if the number of packets to be dropped is absolute (not probability), the longer duration test will make the throughput to be increased since the actual data sent is comparably much more than retransmission overhead.

Special Task: Relationship between protocol parameters and network parameters

Experimental Procedure

1. Configure the server computer to listen with varied window size.
   On the terminal, run: *iperf -s -p 50123 –w##*

2. Configure the client to set the link data rate to be 200MB and the link delay to be 80 msec.
   On the terminal, run: *sudo tc qdisc add dev eth0 root tbf rate 200000kbit latency 50ms burst 5kb*
   On the terminal, run: *sudo tc qdisc add dev eth0 root netem delay 80ms 10ms*

3. Configure the client computer to run for 30 seconds for 30 seconds
   On the terminal, run: *iperf -c 10.10.10.1 -p 50123 -t 30 -w 10000*

4. Observe the throughput of the network.

Experimental Result

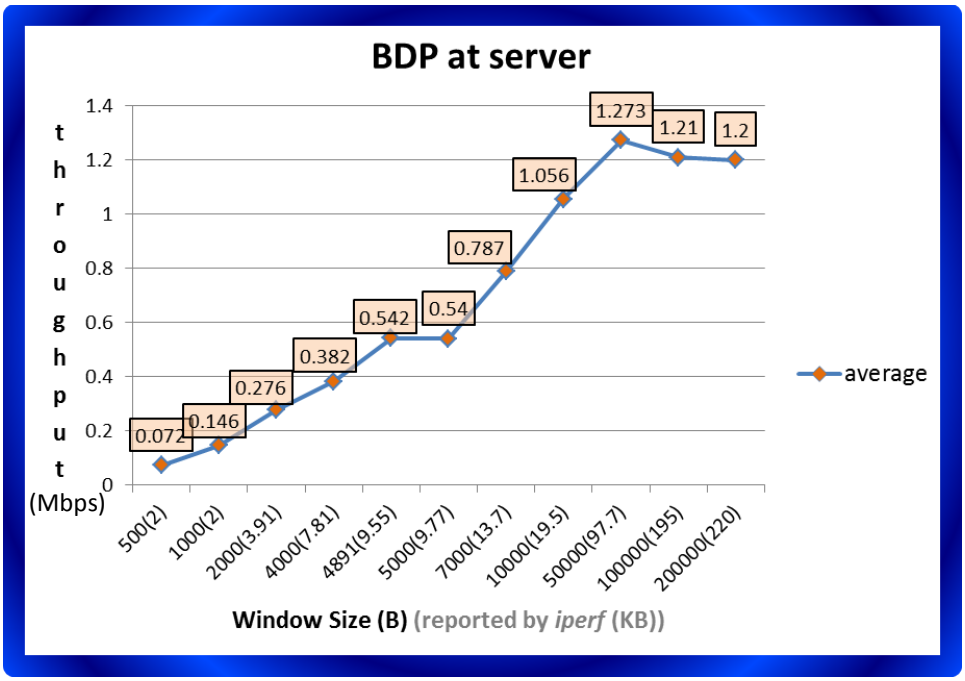

Figure 10: the throughput of the network

Discussion

This experiment focused on the impact of bandwidth-delay product or BDP on the throughput of the network. BDP indicates the amount of data that stayed in the network and can be obtained from the product of data rate and round-trip time. In this case, round-trip time was approximately 80 milliseconds after delay is set to 80 milliseconds and the data rate was approximately 61 Mbps, hence BDP equals to 4880 Mb or 610 MB.

According to Figure 10, throughput of the network slowly increased corresponding to the size of the window. From this experiment, maximum window size was set to be 200 KB which was still less than the BDP since BDP was approximately 610 MB. Hence, there was no effect on the throughput besides continuously increased corresponding to the window size. However, if BDP is smaller than 610 MB, there would be an effect on the throughput in the case where the window size is greater than BDP. The window size would then increase corresponding to BDP instead of window size itself.

In conclusion, both window size and BDP limited the throughput of the network by controlling the amount of data to be transferred before waiting for an ACK within the network. When the window size is less than BDP, throughput would be limited by the window size itself. On the other hand, BDP would limit the throughput in the case where BDP is greater than window size.

Task 3: Multiple TCP sessions

Experimental Procedure

1. Configure the server computer to listen with default configurations.
   On the terminal, run: *iperf -s -p 50123*
2. Configure the client computer to run varied[##] number of TCP sessions for 60 seconds.
   On the terminal, run: *iperf -c 10.10.10.1 -p 50123 -t 60 -P ##*
3. Observe the throughput each TCP session gains and the total network throughput.

Experimental Result

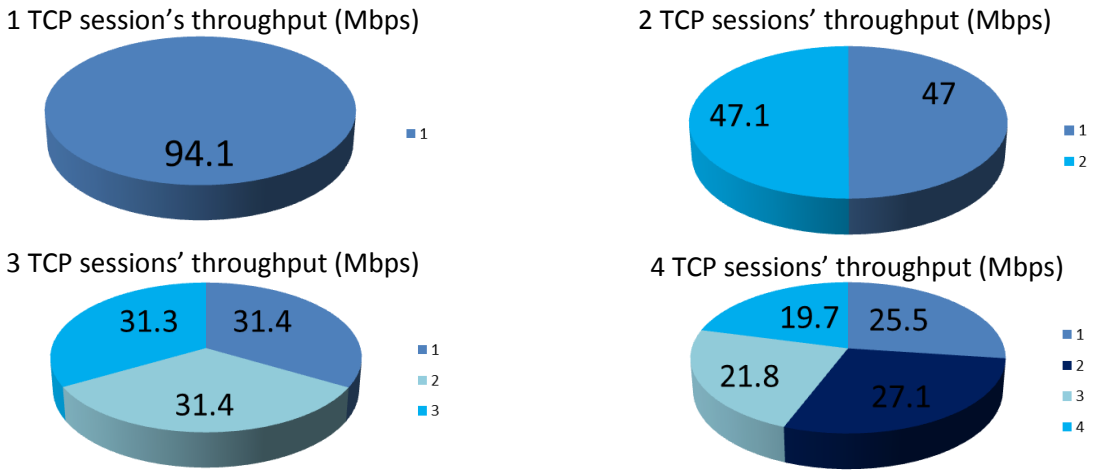

Figure 11: the throughput of each TCP session and the overall network of 94 Mbps

Discussion

According to Figure 11, when there was one running TCP session, this only session gained the throughput of 94 Mbps which was the overall network throughput. However, as the number of sessions in the network increased, each session gained less and less throughput. The overall throughput of the network, which is the summation of each session's throughput, stayed the same at fixed 94 Mbps and was equally allocated to each session.

To conclude, when multiple TCP sessions whose round trip time and segment size are the same were present in the network, TCP protocol treats each session fairly, i.e. tries to allocate equal bandwidth to every session. To illustrate, with n TCP sessions in the network whose capacity is r, each session achieves around r/n throughput. This is known as TCP fairness.

Task 4: Single/Multiple TCP sessions in presence of UDP sessions

Experimental Procedure

1. Configure the server computer to listen both UDP and TCP sessions with default configurations.
   On one terminal (TCP), run: *iperf -s -p 50123*
   On the other terminal (UDP), run: *iperf -s -p 50123 -u*

2. Configure the client computer to run varied## number of TCP sessions with one UDP session for 60 seconds.
   On one terminal (TCP), run: *iperf -c 10.10.10.1 -p 50123 -t 60 -P ##*
   On the other terminal (UDP), run: *iperf -c 10.10.10.1 -p 50123 -t 60 -u –P 1*

3. Observe the throughput each TCP or UDP session gains and the total network throughput.

4. Repeat Step 2 and 3 but change the number of UDP sessions to be two.
   On one terminal (TCP), run: *iperf -c 10.10.10.1 -p 50123 -t 60 -P ##*
   On the other terminal (UDP), run: *iperf -c 10.10.10.1 -p 50123 -t 60 -u –P 2*

Experimental Result

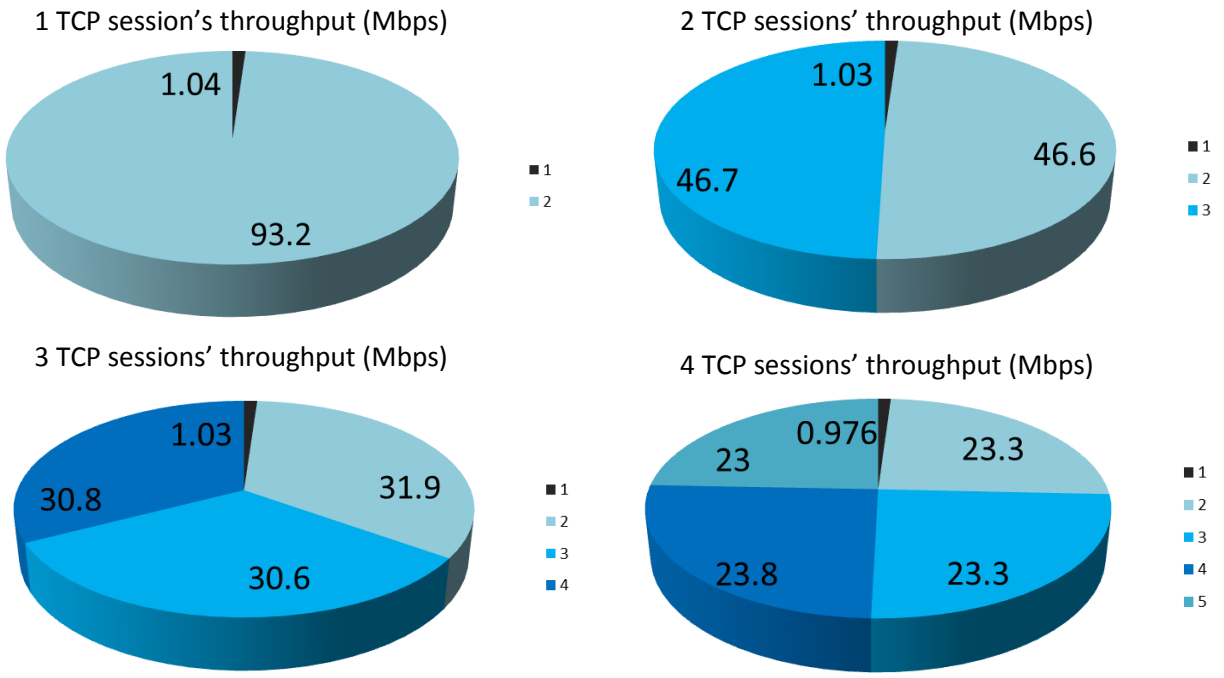

Figure 12: the throughput of each TCP session and the overall network of 94.1 Mbps in presence of one UDP session

1 TCP session's throughput (Mbps) 2 TCP sessions' throughput (Mbps)

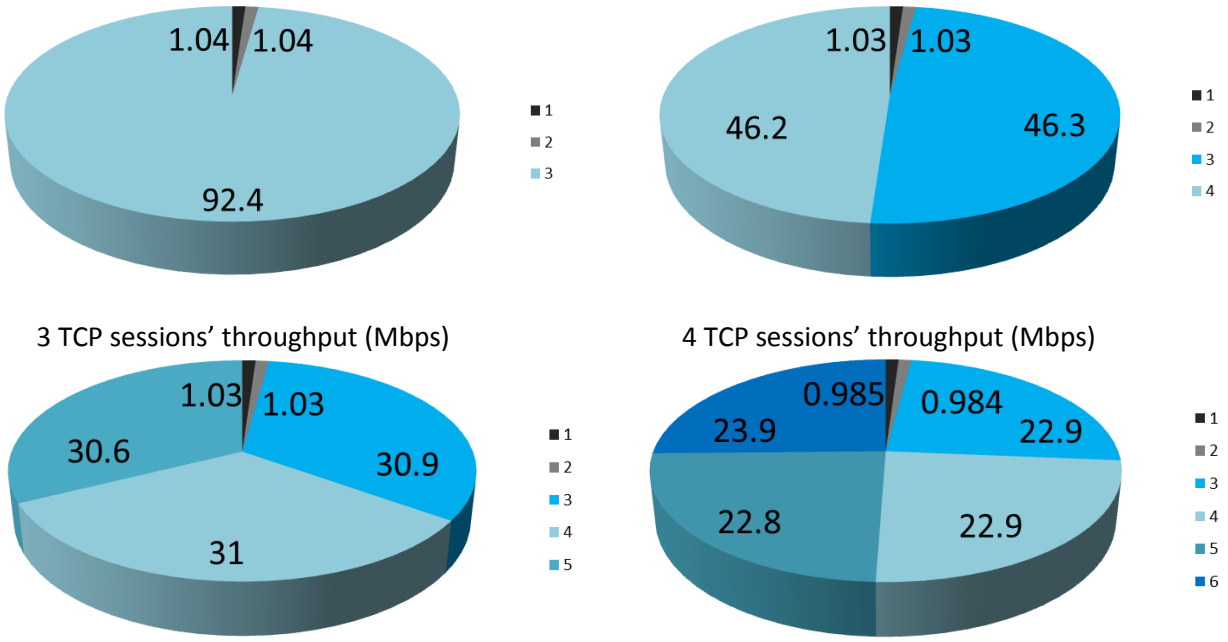

Figure 13: the throughput of each TCP session and the overall network in presence of two UDP sessions

Discussion

According to Figure 12, when there was one UDP session present in the network consisting of varied number of TCP sessions, the TCP sessions were treated with TCP fairness schema as discussed in Task 3. However, the only UDP session always gained the throughput of 1 Mbps regardless of how many additional TCP sessions were introduced into the network while the overall network throughput still stayed fixed at 94 Mbps all the time.

The similar discussion can be drawn from the situation where there were two UDP sessions present in the network. According to Figure 13, each UDP session gained the throughput of 1 Mbps, resulting in the available network throughput left for TCP sessions of 94 Mbps – 2*1 Mbps = 92 Mbps. As seen, this amount of throughput was equally shared among TCP sessions. The result shows that two UDP sessions made each TCP session gain even less throughput than one UDP session in the network.

To conclude, when there are other non-TCP sessions running together in the network with TCP sessions, TCP sessions receive unfair treatment compared to other kinds of sessions. Since TCP fairness does not apply to non-TCP sessions such as UDP ones, only among TCP sessions themselves achieve fairness, which in fact is not the fairness of the whole network. Other non-TCP sessions gain exact throughput they prefer and let TCP sessions share the left throughput. This unfairness can be explicitly seen when there are many more non-TCP sessions exploitatively allocated almost all network throughput while many TCP sessions suffers from the allocation of very little, being-shared-from-what-is-left throughput.

CONCLUSION

The performance of TCP depends on many parameters from the protocol itself and from network environment. The protocol parameters from the experiment are the TCP window size and the TCP data length. The network parameters from the experiment are the link data rate, the link delay and the number of packets dropped. In addition, the hardware specifications of hosts have an impact on TCP performance.