1. Pornprapa Chuwongwit 5122770241
2. Krittameth   Teachasrisaksakul 5122780570
3. Araya   Kinbuangam 5122791171

Test Description

Host Computer Specification

- **Hardware**

Minimum requirements for Ubuntu Desktop (GUI) Installation

- ✓ 1 GHz x86 processor
- ✓ 512MB RAM
- ✓ 5GB of disk space
- ✓ Graphics card and monitor capable of 1024x768

- **Software**

  Ubuntu Desktop is installed to the computer with the aforementioned minimum requirements.

Test Software

1. **Iperf**

Iperf is a prevalent open-source network performance measurement software that can create TCP and UDP data streams, and measure the throughput of a network. Iperf functionalities include a parameter setting for testing a network; a client and server functionality; and measurement of the throughput between the two ends. It runs on various platforms including Linux, Unix and Windows. Install iperf on all host computers used in the test by running the following command line:

```
$ sudo apt-get install iperf
```

2. **Iptables**

Iptables is a built-in firewall on Ubuntu. It allows a user to configure the firewall rules to perform some actions on the input/output/forward chain.

3. **tc**

Tc is a tool for configuring Traffic Control in the Linux kernel. It possesses the following functionality: shaping transmission rate and burst, scheduling (i.e. reordering of packets), policing, dropping etc. Processing of traffic is controlled by three kinds of objects: qdiscs, classes, and filters.

Network Technologies

There are four scenarios to be considered and tested:

1. Single TCP session; varying application/protocol parameters
2. Single TCP session; varying network/link conditions
3. Multiple TCP sessions
4. Single/Multiple TCP sessions in presence of UDP sessions

All scenarios except the second one require the same steps for preparation.

Preparation Steps for Scenario 1, 3, and 4

1. Connect two host computers, with the specifications as stated in ***"Host Computer Specification"*** section, through a cross-over cable according to the *Figure 1.1*

Host A ---------- Host B
Server Client
IP address: a.b.c.j IP address: a.b.c.k

Figure 1.1: Network Topology for Preparation for Scenario 1, 3, and 4

2. According to the *Figure 1.1*, assume an interface ***ethX*** of the server computer A is connected to an interface ***ethY*** of the client computer B through a cross-over cable. Configure the IP address of each computer's interface by the following steps:
    a. To set the IP address *a.b.c.j* to the interface *ethX* on the computer A, run the following command line:

```
$ sudo ifconfig ethX a.b.c.j netmask 255.255.255.0 up
```

    b. To set the IP address *a.b.c.k* to the interface *ethY* on the computer B, run the following command line:

```
$ sudo ifconfig ethY a.b.c.k netmask 255.255.255.0 up
```

3. To test whether the subnet configuration in the previous steps works, perform the following steps:
    a. On the computer A, run the following command line:

```
$ ping a.b.c.k
```

    If the subnet configuration works, the Terminal will periodically display the similar responses to the *Figure 1.2*

```
PING 192.168.1.21 (192.168.1.21): 56 data bytes
64 bytes from 192.168.1.21: icmp_seq=0 ttl=64 time=47.433 ms
64 bytes from 192.168.1.21: icmp_seq=1 ttl=64 time=11.968 ms
64 bytes from 192.168.1.21: icmp_seq=2 ttl=64 time=12.117 ms
64 bytes from 192.168.1.21: icmp_seq=3 ttl=64 time=12.621 ms
--- 192.168.1.21 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 11.968/21.035/47.433/15.243 ms
```

Figure 1.2: Ping Response for Correct Subnet Configuration

    b. On the computer B, perform the same actions as in step 3-a. but run the following command line.

```
$ ping a.b.c.j
```

Test Methodology

Number of Tests

For all scenarios, at least three times of test (step 2 to step 6) must be performed for each different values of each parameter. The additional tests can be performed depending on the difference of the results. If the results differ significantly, three additional tests should be performed. For example, if the values of throughput from repeating the test for three times are 16.7, 17.2, and 17.8, no additional test is required. However, if the values are 17, 20, and 23, the additional tests should be performed.

Test Duration

For scenario 1 and scenario 2, the duration of test should be at least 30 seconds; and, for scenario 3 and scenario 4, the duration of test should be at least 60 seconds. The test duration can be specified in iperf command by using –t option.

Scenario 1: Single TCP session; varying application/protocol parameters

There are four parameters in which the changes may affect TCP's throughput performance:

- TCP receive buffer/window size
- Length of data written/read by application

Each parameter requires the same steps for its test but each of them is different in the options used with iperf command. The following are the steps for test in this scenario.

** For step 2 to step 7, each single test's bandwidth (throughput) values must be recorded.
** For step 3 to step 7, each step must be performed with several different values of parameters as specified in the **"Parameters Values"** section.

Steps for test of impacts from varying "TCP receive window size of server computer" on throughput

1. Open the **Terminal** application of Ubuntu by clicking at the "Applications" button at the upper left corner of desktop and following this path: Accessories > Terminal

2. To check the throughput of the TCP connection with default values of parameters, perform the following actions:
   a. On the server computer A, run the following command in order to activate the server to listen to a connection request:

      ```
      $ iperf -s -t 30
      ```

   b. On the client computer B, run the following command in order to establish a TCP connection to the server computer A with –t option to specify the time to transmit to be 30 seconds (a.b.c.j is an IP address of the server computer A.)

      ```
      $ iperf -c a.b.c.j -t 30
      ```

   *For the test to work, perform step 2-a. immediately before 2-b. since the server needs to be started before the client can connect to it.

3. Record the throughput value shown in the Terminal under the word "Bandwidth"

4. To check the throughput of the TCP connection with other values of ***TCP receive window size*** at the server computer, perform the same actions as the step 2-a. and 2-b. but use these commands instead of command in the step 2-a. and 2-b. respectively.

```
$ iperf -s -w 40K -t 30
$ iperf -c a.b.c.j -t 30
```

According to the above command, ***-w*** option is used to specify TCP receive window size to be 8000 bytes at the server computer A since, on Linux systems, when specifying a TCP buffer size with the -w option, the kernel allocates double as much as indicated.

5. Record the throughput value shown in the Terminal under the word "Bandwidth.

6. Repeat steps 4 and step 5 with different values of window size at the server computer A by changing the value specified after the option -w to be other values as specified in the **"Parameters Values"** section.

Steps for test of impacts from varying "BDP and TCP receive window size" on throughput

** The steps of this test are same as the previous test except that the following steps must be performed before step 1 of the previous test.

A. Set round-trip time to be 160 ms by using the following command to set the delay to be 80 milliseconds (in order to fix the BDP at 2000 KB, or any value that is lower than the default BDP).

```
$ sudo tc qdisc add dev eth0 root netem delay 80ms 0ms
```

Steps for test of impacts from varying "length of data written/read by application at server computer" on throughput

** The steps of this test are same as the previous test except step 4.

4. To check the throughput of the TCP connection with other values of ***length of data written/read by application at server computer A***, perform the same actions as the step 2-a. and 2-b. but use these commands instead of command in the step 2-a. and 2-b. respectively.

```
$ iperf -s -l 8K -t 30
$ iperf -c a.b.c.j -t 30
```

According to the above command, ***-l*** option is used to specify length of data written/read by application to be 8000 bytes at the server computer A.

Steps for test of impacts from varying "length of data written/read by application at client computer" on throughput

** The steps of this test are same as the previous test except step 4.

4. To check the throughput of the TCP connection with other values of ***length of data written/read by application at client computer B***, perform the same actions as the step 2-a. and 2-b. but use these commands instead of command in the step 2-a. and 2-b. respectively.

```
$ iperf -s -l 8K -t 30
```

```
$ iperf -c a.b.c.j -l 8K -t 30
```

According to the above command, ***-l*** option is used to specify length of data written/read by application to be 8000 bytes at the client computer B.

Scenario 2: Single TCP session; varying network/link conditions

There are three conditions of link that may affect the throughput:

- Link data rate
- Link delay
- Packet drop rate

Steps for test of impacts from varying "Link data rate" on throughput

1. Link data rate can be specified at the client computer B by using *tc* command to. The latency is the maximum time interval allowed for a packet to wait, and the burst allows a set of bits to be sent above the specified rate. To add a *'token bucket filter'* that limits the sending rate to 500kbps, run the following command:

```
$ sudo tc qdisc add dev eth0 root tbf rate 500kbit latency 50ms burst 5kb
```

2. To check the throughput of the TCP connection with the specified link delay, perform the following actions:
   a. On the server computer A, run the following command in order to activate the server to listen to a connection request:

```
$ iperf -s -t 30
```

   b. On the client computer B, run the following command in order to establish a TCP connection to the server computer A with –t option to specify the time to transmit to be 30 seconds (a.b.c.j is an IP address of the server computer A.)

```
$ iperf -c a.b.c.j -t 30
```

3. Before changing the link data rates to other values, delete the rule added in step 1 by running the same command as in step 1 but change the word *"add"* to *"del"*.
4. Repeat step 1 to step 4 with different values of link data rate specified after the word *"tbf rate"*.

Steps for test of impacts from varying "Link delay" on throughput

1. Link delay can be implemented at the client computer B by using *tc* command to. To add a random delay of between 90 and 110ms (average 100ms) to every packet that the client computer B sends, run the following command:

```
$ sudo tc qdisc add dev eth0 root netem delay 100ms 10ms
```

2. To check the throughput of the TCP connection with the specified link delay, perform the set of actions as in step 2 of ***Steps for test of impacts from varying "Link data rate" on throughput*** section.
3. Before changing the packet delay to other values, delete the rule added in step 1 by running the same command as in step 1 but change the word *"add"* to *"del"*.
4. Repeat step 1 to step 4 with different values of packet drop rates specified after the word *"delay"*.

Steps for test of impacts from varying "Packet drop rate" on throughput

1. Dropping packets can be implemented at the server computer A by using *iptables* command to set up firewall rules in order to drop the input chain's packets, i.e. packets originated from other host computer and have address of computer A as their destination address. To set up the mentioned firewall rule at the server computer A, run the following command:

```
$ sudo iptables -A INPUT -m statistic --mode random --probability 0.03 -j DROP
```

   This command adds a firewall rule with an action of dropping the packets from input chain with the probability 0.03
2. To check whether the rule is correctly added to the set of firewall rules, run the following command to list all current firewall rules:

```
$ sudo iptables -L
```

3. To check the throughput of the TCP connection with the specified packet drop rate, perform the set of actions as in step 2 of ***Steps for test of impacts from varying "Link data rate" on throughput*** section.
4. Before changing the packet drop rate to other values, delete the rule added in step 1 by running the same command as in step 1 but change –A option to –D option.
5. Repeat step 1 to step 4 with different values of packet drop rates specified after *--probability* option.

Scenario 3: Multiple TCP sessions

There are three conditions examined to observe an impact on TCP's throughput performance:

- Two TCP sessions
- Three TCP sessions
- Four TCP sessions

Each condition requires the same steps for its test but each of them is different in the values for –P option, of iperf command, which is used to specify the number of parallel connections. The following are the steps for test in this scenario.

1. Open the "Terminal" application of Ubuntu
2. To check the throughput of two multiple TCP sessions, perform the following actions:
   a. On the server computer A, run the following command in order to activate the server to listen to a connection request:

```
$ iperf -s -t 60
```

   b. On the client computer B, run the following command in order to establish two TCP connections to the server computer A with –t option to specify the time to transmit to be 60 seconds (a.b.c.j is an IP address of the server computer A.)

```
$ iperf -c a.b.c.j -t 60 -P 2
```

3. To check the throughput of three multiple TCP sessions, perform the same steps as step 2 but change the command at the client computer B to be:

```
$ iperf -c a.b.c.j -t 60 -P 3
```

4. To check the throughput of four multiple TCP sessions, perform the same steps as step 2 but change the command at the client computer B to be:

```
$ iperf -c a.b.c.j -t 60 -P 4
```

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

There are four conditions examined to observe an impact on TCP's throughput performance:

- One TCP session (in presence of one UDP session)
- One TCP session (in presence of two UDP sessions)
- Two TCP sessions (in presence of one UDP session)
- Two TCP sessions (in presence of two UDP sessions)

Each condition requires the same steps for its test but each of them is different in the values for –P option, of iperf command, which is used to specify the number of parallel connections. The following are the steps for test in this scenario.

1. Open two windows of "Terminal" application
2. To check the throughput of one TCP session in presence of one UDP session, perform the following actions:
    a. On the first window of server computer A, run the following command in order to activate the server to listen to a UDP connection request:

    ```
    $ iperf –s -u -t 60
    ```

    b. On the first window of client computer B, run the following command in order to establish a UDP connection to the server computer A with –t option to specify the time to transmit to be 60 seconds (a.b.c.j is an IP address of the server computer A.)

    ```
    $ iperf –c a.b.c.j -u -t 60
    ```

    c. Immediately after the step 2-a. and 2-b., on the second window of server computer A, run the following command in order to activate the server to listen to a TCP connection request:

    ```
    $ iperf –s -t 60
    ```

    d. On the second window of client computer B, run the following command in order to establish a TCP connection to the server computer A with –t option to specify the time to transmit to be 60 seconds (a.b.c.j is an IP address of the server computer A.)

    ```
    $ iperf –c a.b.c.j -t 60
    ```
3. To check the throughput of one TCP session in presence of two UDP sessions, perform the same actions as step 2 but change the command in the step 2-a. to the following command, in order to start two UDP sessions:

    ```
    $ iperf –s -u -t 60 -P 2
    ```
4. To check the throughput of two TCP sessions in presence of one UDP session, perform the same actions as step 2 but change the command in the step 2-c. to the following command, in order to start two TCP sessions:

```
$ iperf –s -t 60 -P 2
```

5. To check the throughput of two TCP sessions in presence of two UDP sessions, perform the same actions as step 2 but change the command in the step 2-a. and the step 2-c. to the following commands, in order to start UDP sessions before starting two TCP sessions:

```
$ iperf -s -u -t 60 -P 2
$ iperf –s -t 60 -P 2
```

Parameter Values

For each test in each scenario, only one parameter value must be changed at a time. For instance, in test for impact of changing TCP buffer size on throughput in scenario1, only the TCP buffer size is changed by using –w option. Subsequently, in test for impact of changing packet drop rate on throughput in scenario 2, only the packet drop rate is changed by adding firewall rules by using iptables.

Especially in scenario 2 where tc is used to change the link data rate and link delay rate, the delete command should be executed to delete the prior value before the add command is run to change the condition to new value. The similar actions should be applied to changing of packet drop rate by using iptables in scenario 2 as well.

Scenario	TCP window size	Length of data written/read by application	Link Delay	Packet Drop Rate	Link Date Rate
Default	85.3 Kbytes(Server) 16.0 Kbytes(Client)	8 Kbytes	100 ms	Zero drop rate	94.1 Mb/s
Scenario1	*	*	Default	Default	Default
Scenario2	Default	Default	**	**	**
Scenario3	Default	Default	Default	Default	Default
Scenario4	Default	Default	Default	Default	Default

Figure 2.1: Default Values of Parameters Categorized by Scenario

**** = Recommended values for Scenario 1***

TCP window size: 2, 4, 6, 8, 10, 12, 14, 16, 20, 40, 60, 85.3, 100 Kbytes
Length of data written/read by application: 4, 6, 8, 10, 12 Kbytes

***** = Recommended values for Scenario 2***

Link delay: 10, 50, 100, 120, 150, 200, 300, 400, 500, 600, 700 Kbytes
Packet drop rate: 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.1, 0.2
Link data rate: 100, 500, 1000, 2000, 3000, 4000, 5000, 6000, 10000 Kbit/s

The recommended values are for the computer that has the default values as in Figure 2.1. These recommended values will cover almost every significant value of throughput. If the computer used for the test does not have the default values as in the Figure 2.1, the experimenter can try to find the appropriate values of parameters by first trying default value and subsequently trying the higher or lower value with equal difference. For example, default window size (85.3 KB) is first used then 60 KB and 100 KB are later set as window size. Afterwards, 40 KB and 120 KB are set as window size. If all throughput values are not yet obtained, experimenter can try more specific value. For

instance, if 20 KB window size yields throughput of 93.93 Mb/s and 10 KB window size yields the throughput of 61.82 Mb/s, the difference between throughput is too large so the experimenter can try the window size of 13, 15, 17 KB to obtain more thorough values of throughput.

Results and Discussions

Scenario 1: Single TCP session; varying application/protocol parameters

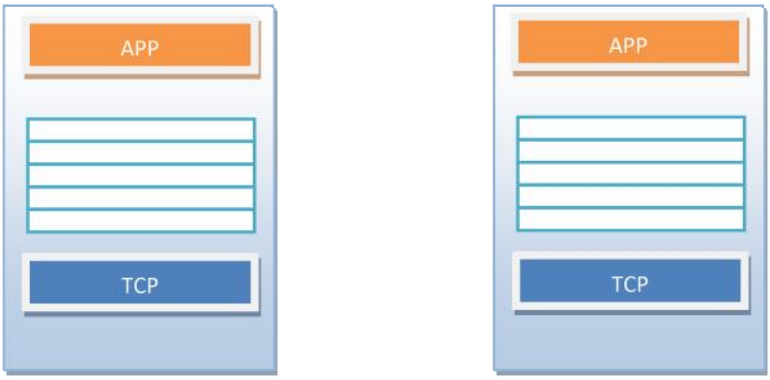

Figure 1.1: TCP Buffer Structure of Server Computer and Client Computer

In the test, the impact of changing application/protocol parameters, i.e. receiver buffer/window size and length of data written or read by the application, on throughput is investigated. The tests are performed by changing only one parameter value at a time. TCP window size is changed by specifying the value after –w option of iperf and length of data written or read by the application is changed by specifying the value after –l option of iperf.

According to the Figure 1.1, the blocks between Transport Layer (TCP) and Application Layer (APP) are the buffer of client computer and server computer. The option "-l" controls the length of data written or read to each sub-block of buffer while the option "-w" controls the overall size of buffer or the total size of five sub-blocks.

TCP Buffer/Window Size

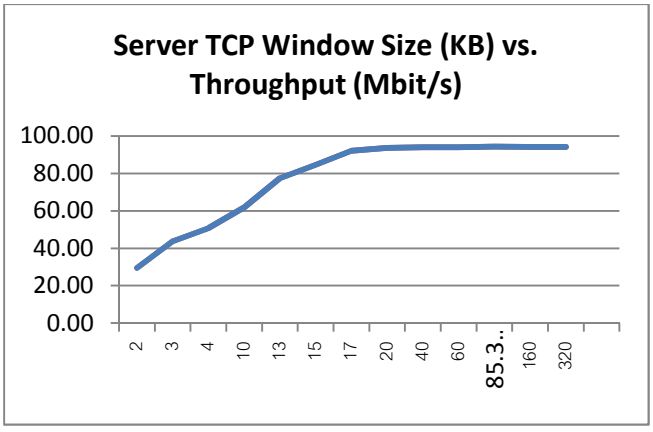

Figure 1.2: Plots of Server's TCP Window Size (Receive Window Size)

According to the Figure 1.2, the throughput value is increased while receive window size is increased. The cause of this trend involves "Flow Control" and "Buffered Transfer" schemes used by TCP. Flow control is for ensuring that the sender's data does not overflow the receiver's buffer by informing the sender about the size of receiver's free buffer space. Whereas, TCP transmits data by using Buffered Transfer scheme: the data created by the application is stored in the buffer until the

total size of the stored data reaches the buffer size. Once the limiting size is reached, the protocol forces the transmission of the packet.

Therefore, TCP receive window size specifies the maximum amount of data that can be transmitted continuously without waiting for the ACK. The larger the window size is the smaller number of times of waiting for ACK and thus reduces delay in total transmission of whole data. As a result of shorter waiting interval, more bytes can be transmitted and the throughput is increased.

Bandwidth Delay Product (BDP) and Receiver Window Size

Two of several factors which influence the throughput are Bandwidth Delay Product (BDP) and receiver window size. BDP is a product of network bandwidth and network round-trip-time (RTT). BDP indicates the network capacity or the amount of data that can be in transit between client and server. If it is less than receiver window size, it will limit the TCP throughput since TCP throughput cannot reach its maximum value. Conversely, if the BDP is large, the receiver window size will limit TCP throughput due to TCP's flow control scheme.

The default link data rate or network bandwidth of tested computers is 100 Mb/s and RTT is 100 millisecond. Default BDP of the tested computer is approximately 1250 kilobytes. (= $100 * 10^6 * 8 * 100 * 10^{-3} / 1000$)

In the test, the BDP value is fixed to be 1000 kilobytes by changing the RTT to be 80 millisecond using tc command and iperf is run with increasing receiver window size in order to investigate the impact of BDP which is lower than window size on throughput.

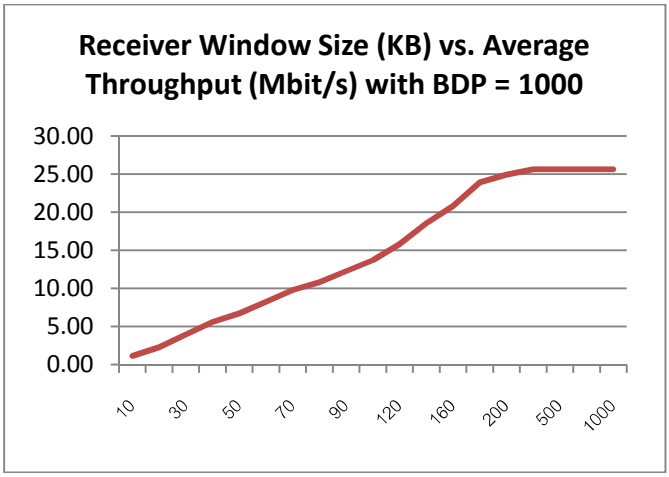

Figure 1.3: Plots of Server's TCP Window Size (Receive Window Size) with 1000 Kilobytes BDP

The Figure 1.2 is when the BDP is 1250 KB and the Figure 1.3 is when the BDP is 1000 KB. According to the Figure 1.2 and Figure 1.3, at the receiver window size of 85 KB, the throughput of default BDP is 94.1 Mb/s while the throughput of 1000 KB BDP is just 5.87 Mb/s. And all throughput values of 1000 KB BDP are lower than those of default BDP since the 1000 KB BDP is less than window size.

Length of Data Written/Read by the Application

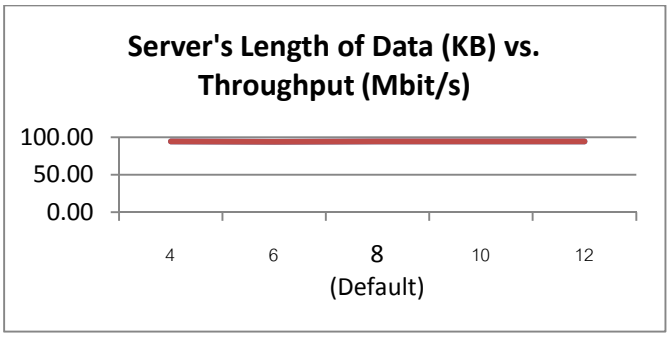

Figure 1.4: Plots of Server's Length of Data Written/ Read by Application and Throughput

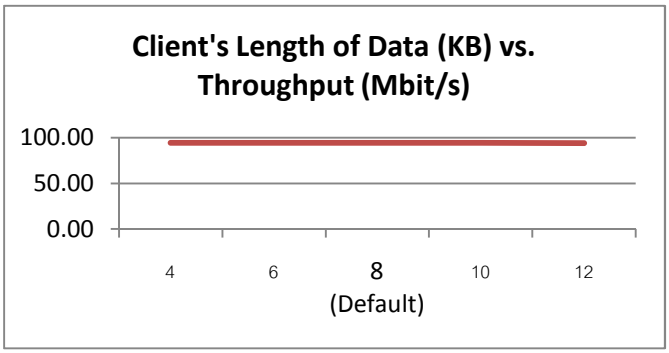

Figure 1.5: Plots of Client's Length of Data Written/ Read by Application and Throughput

According to the Figure 1.4 and Figure 1.5, changes in the length of data read by the server's and written by client's application have no effect on the value of throughput since they merely limit the length of data written to/read from the buffer but do not alter neither the maximum amount of data that can be received without acknowledgement nor the way that the data is segmented into frame. The changes do not alter the way and the speed that the packets are transmitted between a server computer and a client computer. Therefore, the changes do not impact the throughput value.

Scenario 2: Single TCP session; varying network/link conditions

Link Delay

According to the Figure 2.1, the greater link delay at the client computer, set by the tc command to control the queue of packet transmission, yields the lower throughput of TCP connection. The cause of this fact is derived from the lower number of bytes that can be transmitted due to delay. In other word, comparing the situation which the delay rate is high to the situation where there is no delay and the equal time interval is given for both situations, the latter can deliver more bytes than the former since it does not waste any second on waiting for the expected packets to arrive, i.e. delay. The formula for calculating throughput is number of transmitted bytes divided by the total number of seconds using for transmitting those bytes. The lower the number of transmitted bytes yields the lower throughput.

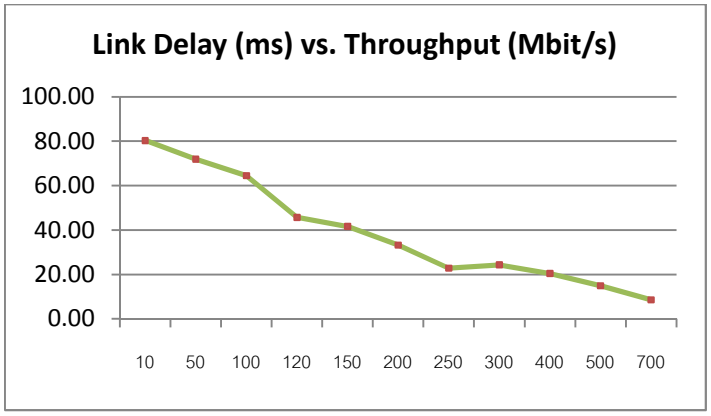

Figure 2.1: Plots of Link Delay and Throughput

Packet Drop Rate

According to the Figure 2.2, setting higher packet drop rate at the server computer, by configuring the firewall rule to drop the input chain's packets, makes the throughput of TCP connection lower. This occurs owing to two reasons. Firstly, TCP is a reliable transport protocol. In other words, the protocol guarantees delivery of packets by using retransmission scheme in case of losing packets. Higher packet drop rate means higher number of lost packets which require TCP to retransmit the packet and as a result waste more time than the situation where lower packet drop rate is implemented. Secondly, TCP uses Congestion Control scheme which reduces the sending rate by half upon detection of a loss event. The reduction of sending rate directly reduces the number of bytes which can be transmitted and thus reduce the throughput.

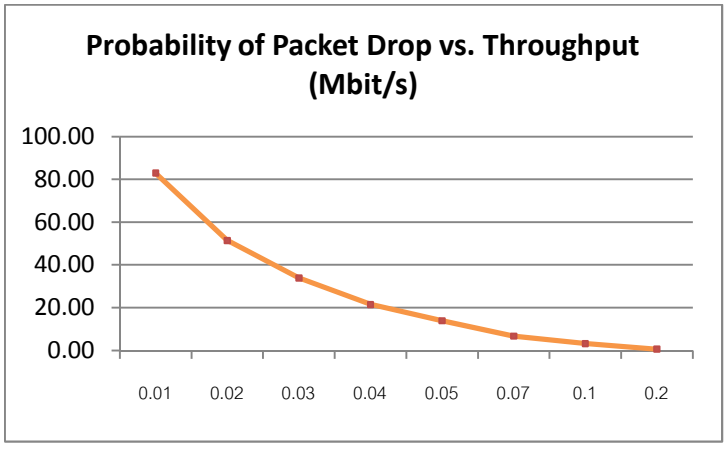

Figure 2.2: Plots of Packet Drop Rate and Throughput Using TCP

Link Data Rate

According to the Figure 2.3, TCP throughput is directly proportional to link data rate. As the link data rate is increased, the throughput is also increased. The reason is that link data rate directly indicates the number of bits that can be delivered per second which directly affects the throughput. Moreover, the Figure 2.3 shows that the trend is almost linearly increasing trend which corresponds to the relatively constant ratio of the throughput and link data rate, approximately 0.95 or 95%.

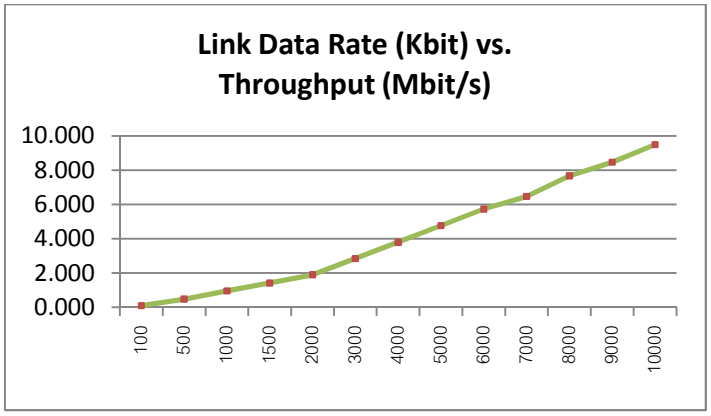

Figure 2.3: Plots of Link Data Rate and Throughput

Scenario 3: Multiple TCP sessions

According to the Figure 3.1, the values of average throughput per TCP session are decreased as the number of sessions is increased, and their values, 47.72, 31.42, and 18.83 Mb/s, are approximately the half, one-third, and one-fourth of total average throughput, respectively. This indicates TCP throughput performance is shared among multiple TCP sessions and each session gets lower throughput when the number of TCP session is higher. However, the higher number of TCP sessions does not have any impact on the total average throughput with one TCP session, i.e. 94.1 Mb/s.

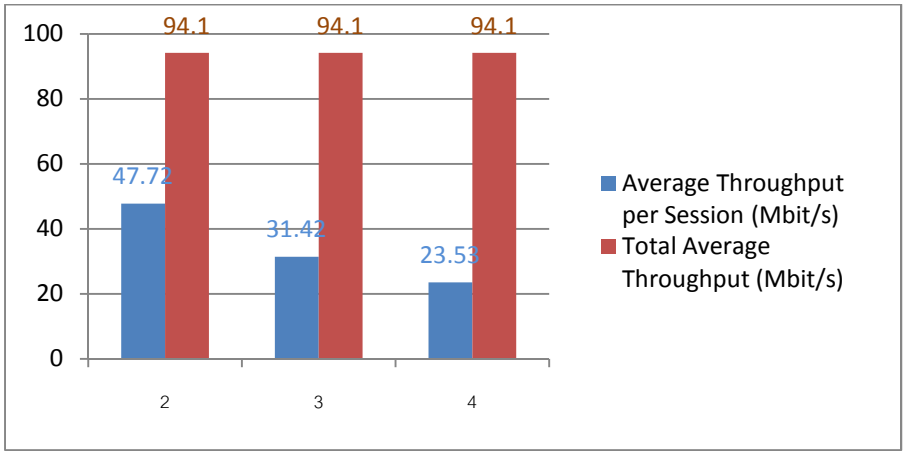

Figure 3.1: Bar Graph of Number of TCP Sessions and Average Throughput per Session and Average Throughput of Multiple TCP Sessions

Each TCP session gets a fair share of throughput. According to the Figure 3.2, the differences between the maximum throughput per session and the average throughput; and between the minimum throughput per session and the average throughput are about 1 and 5 Mb/s for two and three TCP sessions, respectively. This occurs since TCP ensures fairness of performance share among multiple TCP connection.

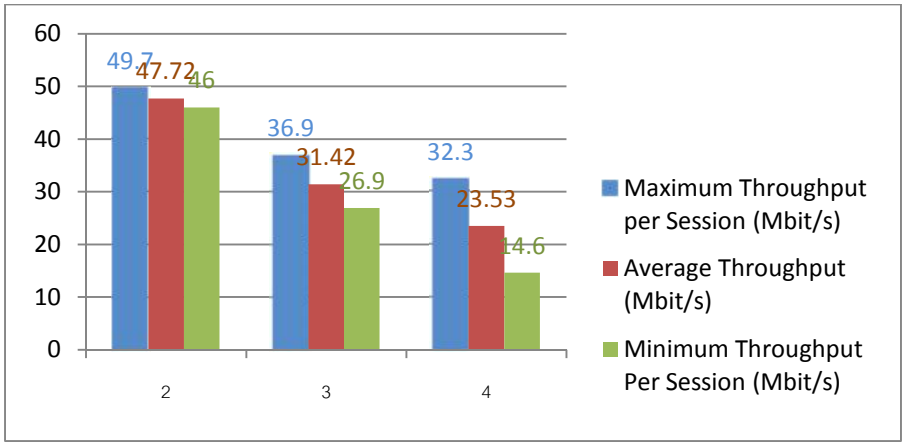

Figure 3.2: Bar Graph of Number of TCP Sessions, Maximum Throughput Per Session, and Minimum Throughput Per Session

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

Single TCP Session in Presence of Single/Multiple UDP Sessions

According to the Figure 4.1, the value of throughput where default values of parameters are applied and no UDP session is present is 94.1 Mb/s. TCP throughput is 93.2 Mb/s (reduced by 0.90 Mb/s) in presence of one UDP session and 92.1 Mb/s (reduced by 2.00 Mb/s) in presence of two UDP sessions. These reductions in TCP throughput values by multiples of 1 Mb/s, which is close to the value of each UDP session's throughput (i.e. 1.05 Mb/s), indicate that, in presence of UDP session, the throughput is shared between the TCP session and the UDP session.

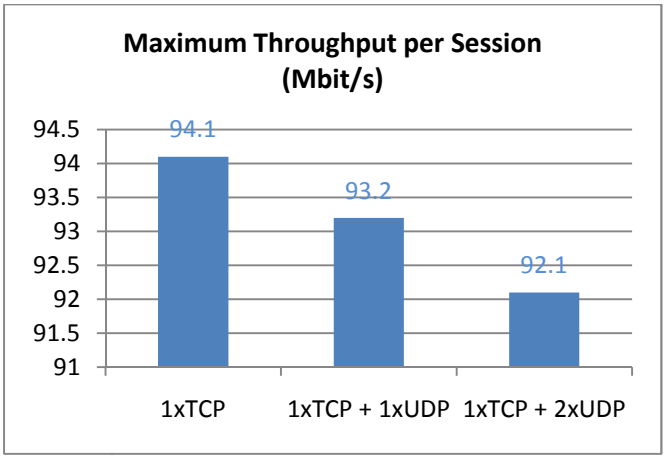

Figure 4.1: Bar Graph of Average Throughput of One TCP Session in Presence of Single/Multiple UDP Sessions

Multiple TCP Sessions in Presence of Single/Multiple UDP Sessions

According to the Figure 4.2, the throughput values of two TCP sessions in presence of one UDP session (93.1 Mb/s) and two UCP sessions (92.1 Mb/s) are lower than that of only single TCP session (94.1 Mb/s) since the throughput is shared among two TCP sessions and UDP sessions. These differences are multiples of 1 Mb/s, the throughput of one UDP session.

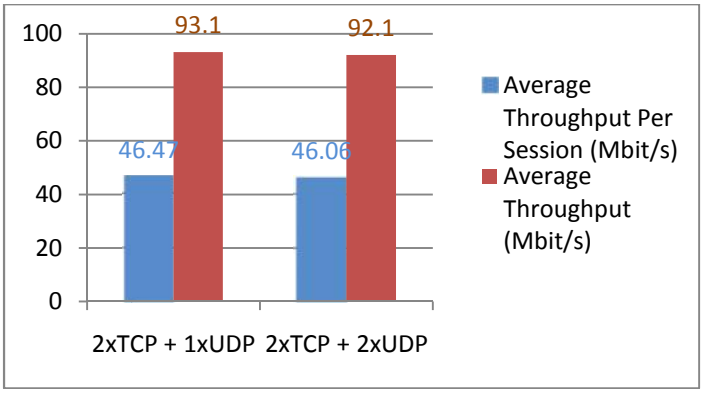

Figure 4.2: Bar Graph of Average Throughput per Session and Average Throughput of Two TCP Sessions in Presence of Single/Multiple UDP Sessions

Appendix

Section 1: Throughput of Varying Application/Protocol Parameters (Scenario 1)

TCP receive buffer/window size of server computer (KBytes)	Throughput(Mbit/s)
2	29.50
3	43.60
4	50.77
10	61.82
13	77.40
15	84.50
17	92.00
20	93.60
40	93.93
60	94.00
85.3 (Default)	**94.30**
160	94.13
320	94.17

TCP receive buffer/window size of client computer (KBytes)	Throughput(Mbit/s)
1	62.47
2	62.60
4	62.60
5	62.50
5.5	91.60
6	91.80
8	92.83
16 (Default)	94.30
24	94.03
32	94.03

Length of Data Written/Read by Application of Server (KBytes)	Throughput(Mbit/s)
4	94.17
6	94.10
8 (Default)	**94.23**
10	94.20
12	94.13

Length of Data Written/Read by Application of Client (KBytes)	Throughput(Mbit/s)
4	94.27
6	94.23
8 (Default)	**94.20**
10	94.17
12	94.10

BDP (1000 KBytes)	
Receiver Window Size (Server)	Average Throughput
10	1.14
20	2.28
30	3.94
40	5.57
50	6.72
60	8.25
70	9.77
80	10.80
90	12.25
100	13.70
120	15.80
140	18.50
160	20.80
180	23.90
200	24.90
300	25.60
500	25.60
700	25.60
1000	25.60

Section 2: Throughput of Varying Network/Link Conditions (Scenario 2)

Link Data Rate		
Link Data Rate (kilobits)	Throughput (Mbits/s)	Percentage
100	0.095	95.00%
500	0.476	95.20%
1000	0.954	95.40%
1500	1.417	94.47%
2000	1.897	94.85%
3000	2.840	94.67%
4000	3.790	94.75%
5000	4.767	95.34%
6000	5.730	95.50%
7000	6.467	92.39%
8000	7.667	95.84%
9000	8.463	94.03%
10000	9.490	94.90%

Link Delay	
Link Delay Interval (ms)	Throughput (Mbits/s)
10	80.32
50	71.90
100	64.48
120	45.70
150	41.62
200	33.23
250	22.85
300	24.33
400	20.43
500	14.90
700	8.57

Packet Drop Rate	
Probability of Dropping Packet	Throughput (Mbits/s)
0.01	82.95
0.02	51.42
0.03	33.88
0.04	21.45
0.05	13.85
0.07	6.71
0.1	3.21
0.2	0.61

Section 3: Throughput of Multiple TCP Sessions (Scenario 3)

2xTCP (Two TCP Sessions)			
Attempt	**Session**	**Throughput Per Session (Mbit/s)**	**Total Throughput (Mbit/s)**
1	1	47.6	94.1
	2	48.2	
2	1	46.6	94.1
	2	48.2	
3	1	46.0	94.1
	2	49.7	
max throughput/session		49.7	
min throughput/session		46.0	

3xTCP (Three TCP Sessions)			
Attempt	**Session**	**Throughput Per Session (Mbit/s)**	**Total Throughput (Mbit/s)**
1	1	32.1	94.1
	2	34.5	
	3	27.8	
2	1	33.1	94.1
	2	27.8	
	3	33.2	
3	1	32.0	94.1
	2	31.4	
	3	31.1	
4	1	29.5	94.1
	2	36.9	
	3	27.8	
5	1	36.8	94.1
	2	26.9	
	3	30.4	
max throughput/session		36.9	
min throughput/session		26.9	

4xTCP (Four TCP Sessions)			
Attempt	**Session**	**Throughput Per Session (Mbit/s)**	**Total Throughput (Mbit/s)**
1	1	32.3	94.1
	2	21.8	
	3	14.6	
	4	25.5	
2	1	23.5	94.1
	2	25.7	
	3	20.6	
	4	24.3	
3	1	26.6	94.1
	2	28.9	
	3	19.3	
	4	19.3	
max throughput/session		32.3	
min throughput/session		14.6	

Section 4: Throughput of Single/Multiple TCP Sessions in Presence of UDP Sessions (Scenario 4)

1xTCP + 1xUDP	
Attempt	Throughput (Mbit/s)
1	93.2
2	93.2
3	93.2

1xTCP + 2xUDP	
Attempt	Throughput (Mbit/s)
1	92.1
2	92.1
3	92.1

2xTCP + 1xUDP			
Attempt	Session	Throughput (Mbit/s)	Total Throughput (Mbit/s)
1	1	45.6	93.1
	2	47.5	
2	1	48.5	93.1
	2	44.6	
3	1	49.2	93.1
	2	42.9	
4	1	46.3	93.1
	2	46.8	
5	1	42.5	93.1
	2	50.6	
6	1	47.9	93.1
	2	45.2	

2xTCP + 2xUDP			
Attempt	Session	Throughput (Mbit/s)	Total Throughput (Mbit/s)
1	1	45.7	92.1
	2	46.4	
2	1	48.3	92.1
	2	43.9	
3	1	48.4	92.1
	2	43.7	
4	1	48.6	92.1
	2	43.5	
5	1	46.1	92.1
	2	46.0	
6	1	53.3	92.1
	2	38.8	