Goutam Bhat – *goubh275*
Santiago Pagola - *sanpa993*

Lab 3 – Transmission Control Protocol (TCP)

Table of Contents

Task A: TCP Basics

The client computer makes a HTTP POST **request** to *gaia.cs.umass.edu*. To transmit the request, a TCP connection is setup between port 1161 of the client computer and port 80 of the server (*gaia.cs.umass.edu*). Since the HTTP request is too large to be transmitted as a single TCP segment, it is broken down into multiple chunks (122).

To set up the connection, a three way handshake is performed (shown in packets 1 to 4 in the Wireshark packet list) between the client (192.168.1.102:1161) and the server (128.119.245.12:80). After this the client keeps sending the TCP segments. The client also keeps an estimate of the RTT using the time elapsed between the sending of a segment and the receipt of acknowledgement for the same. This estimate of the RTT is used to determine packet drops. If the client doesn't receive any acknowledgement for a packet within a certain time (depending on the estimated RTT), the client interprets that the packet is lost and hence retransmits it.

Goutam Bhat – *goubh275*
Santiago Pagola - *sanpa993*

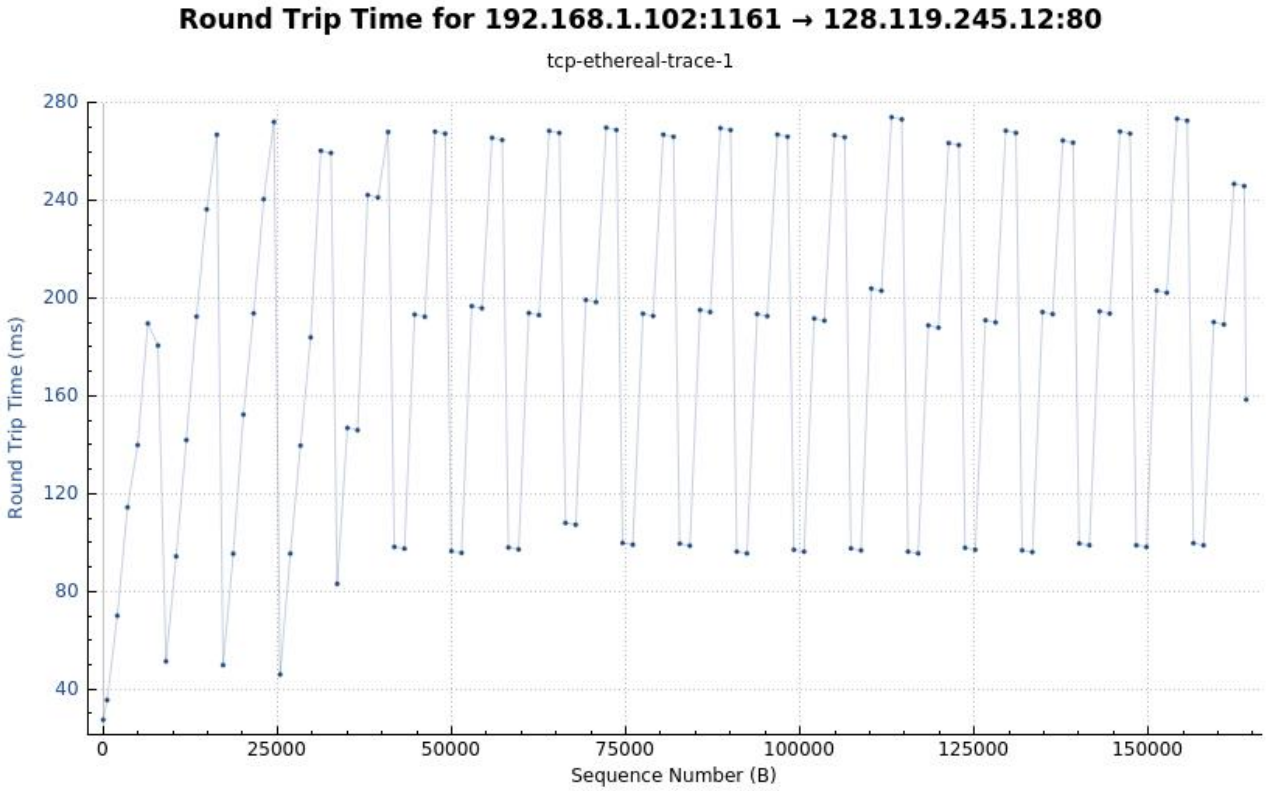

Illustration 1: The figure above shows an RTT vs. Sequence number plot of the whole file upload.

Observations

- In order to determine whether a packet is of SYN / SYN-ACK / ACK type, the field ‘Flags’ in the TCP packet header is updated accordingly, e.g., if the packet represents a SYN header, the corresponding SYN bit is enabled, and the rest of the flag-bits are set to 0.
- The length of the first TCP segment is 565 bytes, and the length of the following 5 is 1460 bytes, which is the recommended MSS for TCP.

Goutam Bhat – *goubh275*
Santiago Pagola - *sanpa993*

Task B: TCP Congestion Control in Action

1. The TCP's *slow start* phase's end is not possible to spot from the graphs. It seems that the data is coming from the host browser to the TCP layer at a lower rate than then network capacity, so the TCP layer does not have to aggressively increase its *cwnd*. The client sends batches of 6 segments, and seems to wait for the acknowledgement for all these segments. If the data was continuously available at the sender buffer, then at least a new segment should have been sent upon the receipt of an acknowledgement.

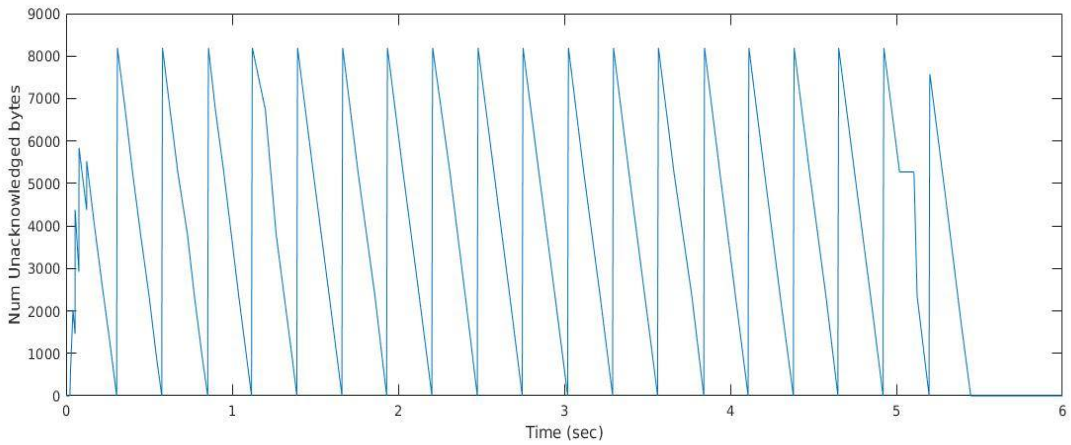

Illustration 2: Plot of Number of unacknowledged bytes vs time. It can be seen that the client sends 6 packets, waits for the acknowledgements to arrive for these packets. Once it has got all the acknowledgements, it sends the next batch of packets.

2. The congestion window (*cwnd*) is maintained at the client, and denotes how many unacknowledged segments there can be in the network. This is maintained so as to not congest the network by sending too many bytes at once.

   The receive window (rwnd) is sent by the receiver and denotes how much more data the receiver can buffer. This is done for flow control, i.e. to ensure that sender doesn't send data at a higher rate than the receiver can handle.

   The number of unacknowledged bytes is used by the sender for both flow control as well as congestion control. It should be lower than both *cwnd* and *rwnd*.

   The effective window (*EW*) at the sender denotes how many unacknowledged bytes the sender can have in the network.

$$EW = last\ byte\ sent - last\ byte\ acknowledged \leq min(cwnd, rwnd)$$

Formula 1: Effective window inequality

3. If the host is sending the data at a high rate so that the network speed is the bottleneck, then it should be possible to find the *cwnd* from the trace by looking at how many unacknowledged bytes there are in the network, since that should be equal to *cwnd* when the TCP sender always has data to send. However if the sender is sending data at low rate, then its not possible.

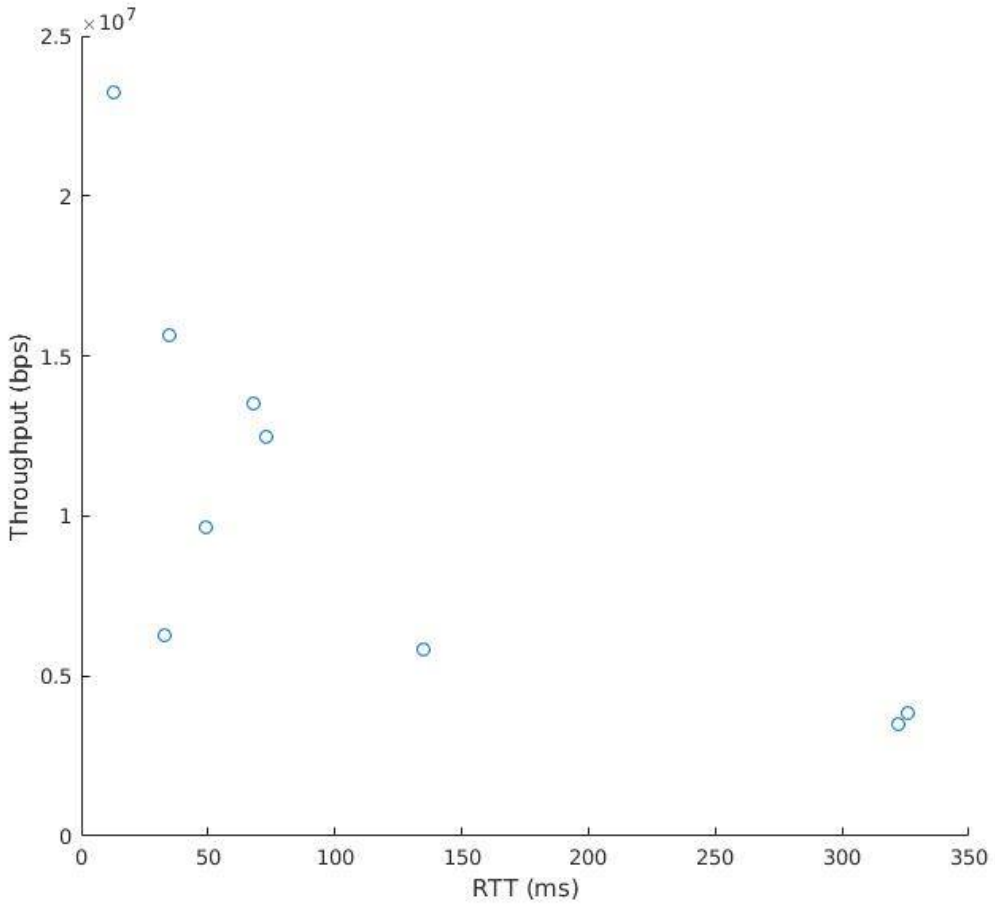

Illustration 3: RTT (in milliseconds) vs. Throughput (in bps) for task c2. It can be noticed that the lower the RTT value, the higher the achieved throughput.

Observations

- The reason why the 3-way handshake is done in the first 3 packets is because when the client sends the first SYN packet, the SYN-ACK response that the client expects from the server times out (it actually comes in packet number 6, marked in black as “Out-Of-Order”.), so the client retransmit the SYN packet.

Goutam Bhat – *goubh275*
Santiago Pagola - *sanpa993*

Task C: A Short Study of TCP Fairness

The throughput for every connection has been calculated by simply dividing the number of transferred bytes (converted to bits) with the amount of time, i.e.,

$$TP(bps) = \frac{number\ of\ transferred\ bytes * \frac{8\,bits}{1\,byte}}{time(s)}$$

Formula 2: Calculation for the throughput of a given TCP connection

1. Case: Four clients, four downloads, same host, same server

The following formula is well-known for estimating the steady-state throughput of a TCP connection:

$$Average\ throughput\ of\ a\ connection = \frac{1.22 * MSS}{RTT * \sqrt{L}}$$

Formula 3: Average throughput of a connection, depending on the MSS, RTT and L

, with MSS being the Maximum segment size, RTT the round-trip time and L the loss rate. Assuming an equal loss rate for the same connections sharing the same bottleneck link, we establish the simplified relationship as:

$$Average\ throughput\ of\ a\ connection \sim k * \frac{1}{RTT}$$

Formula 4: Simplification of Formula 2 for a scenario where all connections share the same link, thus having the same losses L

, meaning that the higher the RTT, the lower the average throughput of the connection, i.e., RTT and throughput are inversely proportional.

Since all clients (within the same host) are using the same link to retrieve the files, in this particular case RTT can be “ruled out” from the equation and the total bandwidth of the host can be obtained by adding up the individual bandwidths of every connection. This points out TCP’s fairness.

Goutam Bhat – *goubh275*
Santiago Pagola - *sanpa993*

Connection	Total transferred bytes	Duration (s)	RTT (ms)	Throughput (bps)
1	165095720	521	12	2535059
2	165842766	521	12	2546529
3	165458792	514	12	2575234
4	163235772	512	12	2550558
			Total:	**10207380**

2. Case: One client, one download, one host, different mirror servers

As in Formula 2, the total bandwidth has been calculated by adding up the individual bandwidths of every connection made. Since every connection used a different link – all mirrors are different - , little can be commented on the TCP fairness. The only thing that can be pointed out from the results is that in general, lower the RTT, higher the throughput, though there seem to be some exceptions.

Connection	Total transferred bytes	Duration (s)	RTT (ms)	Throughput (bps)
1	261319130	90	13	23228367
2	175995832	90	35	15644073
3	151894552	90	68	13501737
4	140388568	90	73	12478983
5	108610702	90	49	9654284
6	70644690	90	33	6279528
7	65744938	90	135	5843994
8	43212876	90	326	3841144
9	39222524	90	322	3486446
			Total:	**93958556**

Goutam Bhat – *goubh275*
Santiago Pagola - *sanpa993*

3. Case: BitTorrent download from multiple peers

Once again, as in Formula 2, the total bandwidth has been calculated by adding up the individual bandwidths of every connection. Again since the connections might use totally different links, it's hard to comment on TCP fairness.

Connection	**Total transferred bytes**	**Duration(s)**	**RTT(ms)**	**Throughput (bps)**
1	108851134	58	40	15013949
2	90435681	58	36	12473887
3	57971584	53	100	8750427
4	32000012	29	68	8827589
5	32557334	35	31	7441676
6	27199361	31	33	7019189
7	26329578	31	122	6794729
8	38834490	56	146	5547784
9	23571761	35	74	5387831
10	36252962	55	66	5273158
			Total:	**82530219**

Observations

Both in cases 2 and three, since all the connections are using different links, each of which may have different losses, it's hard to comment on TCP's fairness, though in general, a lower RTT means that the throughput will be higher, assumming similar losses in all links.

Goutam Bhat – *goubh275*
Santiago Pagola - *sanpa993*

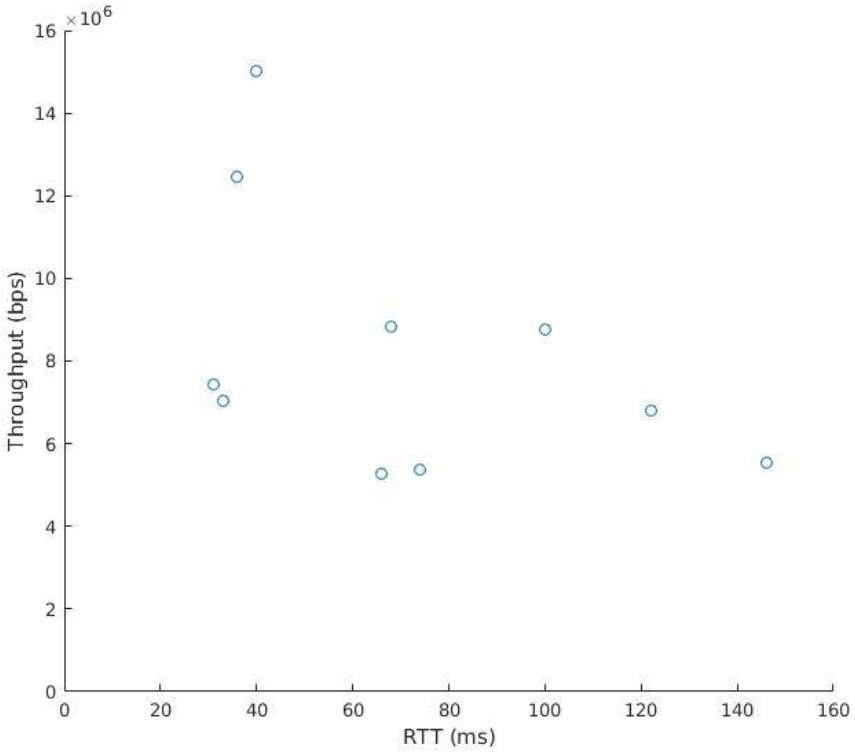

Illustration 4: RTT (in milliseconds) vs Throughput (in bps) plot for task c3. As in Illustration 3, it can be seen that the lower the value of the RTT, the higher the throughput.