



TDTS04 Computer Networks and Distributed Systems

TCP Lab

Authors

Maximilian Bragazzi Ihrén, maxbr431@student.liu.se

Markus Lewin, marle943@student.liu.se



VT1
Version 1.0

1. Audit Descriptions

Ver.	Audit Description	Date
1.0	Written for report.	150220

2. Task A

1. What are the first and last packets for the POST request?

The first packet is packet #4 (if you do not count the handshake, otherwise it is packet #1).

The last packet is packet #197 (not counting ACK-s, in which case it is packet #202).

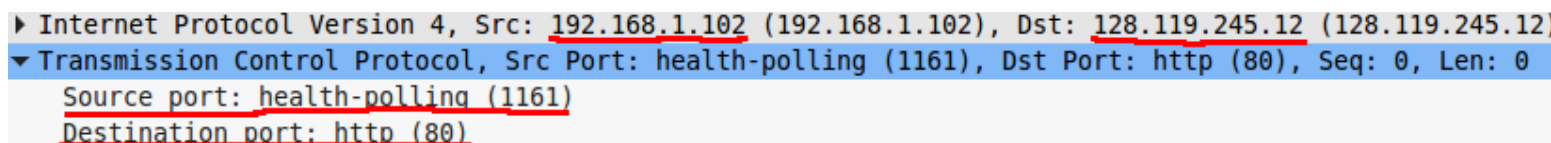
2. What is the IP address and the TCP port used by the client computer (source) that is transferring the file to gaia.cs.umass.edu?

IP: 192.168.1.102

Source port: 1161

Dest port: 80

See figure 1.



▶ Internet Protocol Version 4, Src: 192.168.1.102 (192.168.1.102), Dst: 128.119.245.12 (128.119.245.12)
▼ Transmission Control Protocol, Src Port: health-polling (1161), Dst Port: http (80), Seq: 0, Len: 0
Source port: health-polling (1161)
Destination port: http (80)

Figure 1

3. What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connections?

IP: 128.119.245.12

Source port: 80

Dest port: 1161

This can be checked in a similar way to the prior question, but for the next packet.

4. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?

Sequence number: 0

The SYN flag under the "flags" part of the packet identifies it as a SYN segment.

See figure 2.

```

.... ..0. .... = Urgent: Not set
.... ...1 .... = Acknowledgment: Set
.... .... 0... = Push: Not set
.... .... .0.. = Reset: Not set
.... .... ..1. = Syn: Set
          A = Fin: Not set

```

Figure 2

5. What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the ACKnowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK segment?

Sequence number: 0

Ack value: 1 (length of recieved packet + seq)

The SYN and ACK flags indicate that the segment is a SYNACK segment (see figure 2).

6. What is the sequence number of the TCP segment containing the HTTP POST command?

Packet 4 (the first TCP data packet).

See figure 3.

```

4 0.026477 192.168.1.10
5 0.041737 192.168.1.10
18 .....P.. ..4.t.P.
65 Dp....P0 ST /ethe
31 real-lab s/lab3-1
2f -reply.h tm HTTP/

```

Figure 3

7. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received? Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the RTT value for each of the six segments? What is the EstimatedRTT value (see page 277 in text) after the receipt of each ACK? Assume that the value of the EstimatedRTT is equal to the measured RTT for the first segment, and then is computed using the EstimatedRTT equation on page 277 for all subsequent segments.

Packet number	Seq number	Send time (seconds)	ACK time (seconds)	RTT (seconds)	EstRTT (seconds)
1	1	0.026477	0.053937	0.02746	0.02746
2	566	0.041737	0.077294	0.035557	0.028472125

3	2026	0.054026	0.124085	0.070059	0.0336704844
4	3486	0.054690	0.169118	0.114428	0.0437651738
5	4946	0.077405	0.217299	0.139894	0.0557812771
6	6406	0.078157	0.267802	0.189645	0.0725142425

RTT being calculated by taking the ACK time and removing the send time (ACK time – Send time).

Estimated RTT being calculated by the following equation:

$$(1 - a) * \text{OldRTT} + a * (\text{ACK_rcv_time} - \text{pkt_snt_time})$$

Where $a = 0.125$.

8. What is the length of each of the first six TCP segments?

Packet number	Length of packet	Length of data
1	619	565
2	1514	1460
3	1514	1460
4	1514	1460
5	1514	1460
6	1514	1460

9. What is the minimum amount of available buffer space advertised at the receiver for the entire trace? Does the lack of receiver buffer space ever throttle the sender?

The starting buffer space is advertised as 5840, and grows up until 62780. The sender is never throttled by this.

10. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?

There are no retransmitted segments in the trace file. We checked this for missing/late ACK values, but everything was received in order.

The last ACK-packet is however “out of order” by checking the ACK-value alone, but this is the ACK for the “200 OK” message that is received, so this was not taken into account.

11. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 on page 285 in the text).

It usually ACKs 1460 bytes at each packet (the bytes received by the packet the ACK is meant for).

And yes, there is at least one case of ACKing multiple packets at packet #61 in the trace. The ACK increase from the prior ACK is 2920, which is 1460 multiplied by 2.

12. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

Throughput equation:

TCP-size (in bits) / latency (in seconds)

TCP-size equation:

Ack-value of last Ack-packet * 8

Latency equation:

Last TCP packet time – first TCP packet time

TCP-size = $164091 * 8 = 1312782$ bits.

Latency = $5.455830 - 0.026477 = 5.429353$ seconds.

Throughput = $1312782 / 5.429353 = 241793.4512638983$ bps

Throughput (approx) = 242 Kbit/s

Discussion:

The ACK-value represents the amount of bytes the receiver has received. This is useful when determining when a packet arrived fully, what size it had, and if some packets were lost one can resend the packet(s) corresponding to that ACK-value.

The buffer at the receiving end's size holds the entire data of the packet, not counting the header information, nor other options that might be attached. It is important that the sending party does not overflow the buffer, as that would lead to packet losses.

RTT is useful when calculating whether your packet took the optimal route, speed-wise. Of course, this might differ as the cost of using one wire might change drastically, which would lead to packets taking a slightly slower route.

RTT is one of the factors you would use to determine what the optimal route the packet could take would be. This, along with the longest matching prefix determination results (most of the time) in the most optimal route. As mentioned above, which link the packet would be sent along is also determined by the cost of using that wire.

If a packet is lost on the way to the receiver, the receiver would notice that something is wrong when it sees that the next packet is in the wrong order. It then sends the previous ACK again, which results in a resend. This could mean wasting data usage, if the prior packet was taking a slower route, and therefore is received later than the first, at which point the receiver sends an ACK for both of the packets combined.

It's possible to determine if a packet has been lost or not (when looking at, for example, a Wireshark trace) by simply looking at the ACKs received and comparing the values to the prior packets. If something seems amiss, a packet has most likely been lost.

3. Task B

13. Use the Time-Sequence-Graph (Stevens) plotting tool to view the sequence number versus time plot of segments being sent from the client to the server (Figure 2a and Figure 2b). For each of the two traces, can you identify where TCP's *slow start* phase begins and ends, and where *congestion avoidance* takes over? To better identify these phases, you may need to find the number of unacknowledged packets (or bytes) at different times and plot the

unacknowledged packets (y-axis) as a function of time (x-axis). Note that the number of unacknowledged packets at different times can be found by comparing the number of packets that have been sent with the number of packets that have been acknowledged. After plotting the number of unacknowledged packets versus time, comment on ways in which the measured data differs from the idealized behavior of TCP that we've studied in the text.

The first trace's slow start can be seen in figure 4.

There does however not seem to be any congestion control in the first trace. There could be, but there are no defining characteristics.

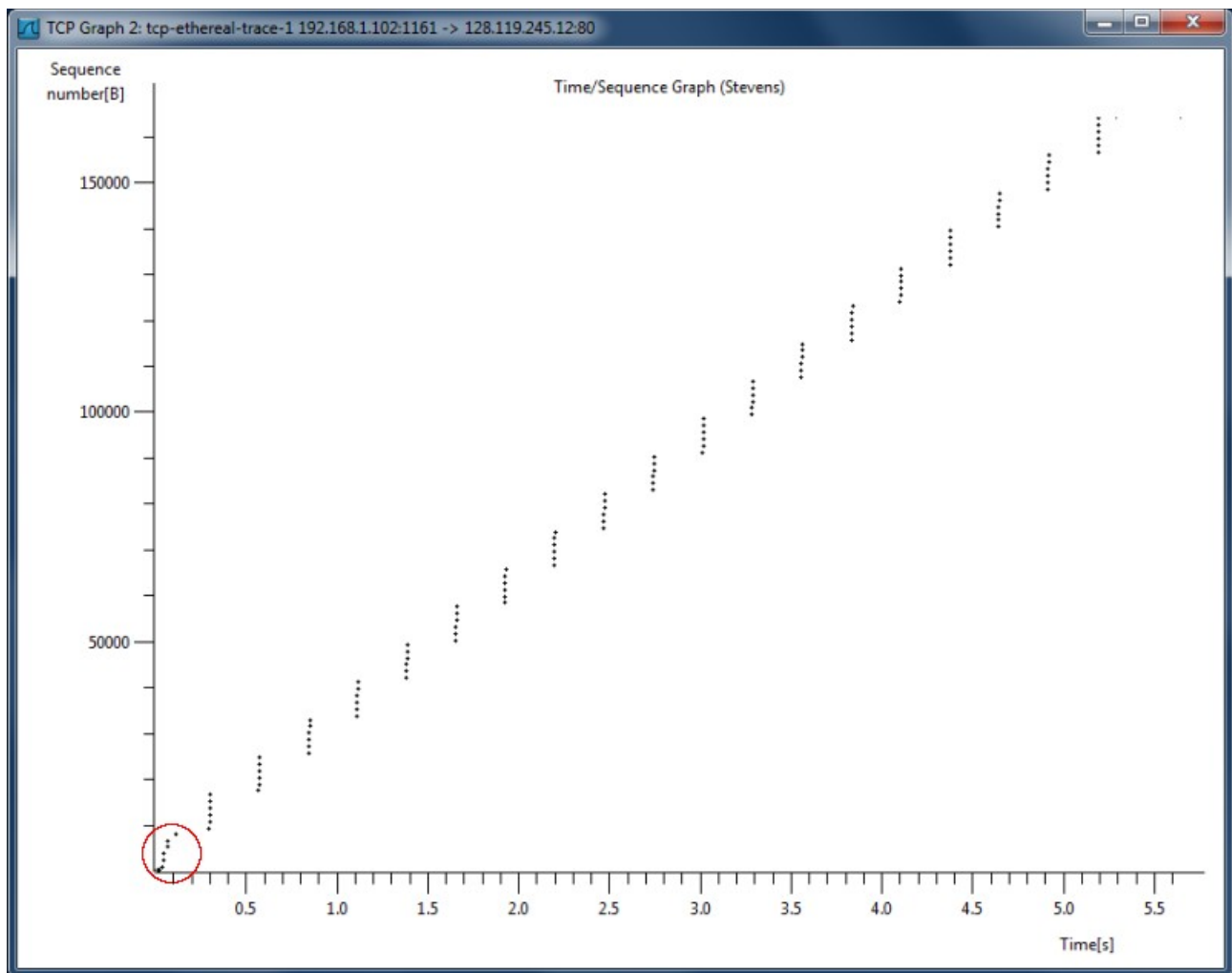


Figure 4.

The second trace's slow start can be seen in figure 5.

The slow start in the second trace is seemingly the straight line in all of the "segments", which later go into congestion control (which would be the diagonal line). However, it seems like after each diagonal line, it goes into slow start once again. This could be due to a timeout error from a packet.

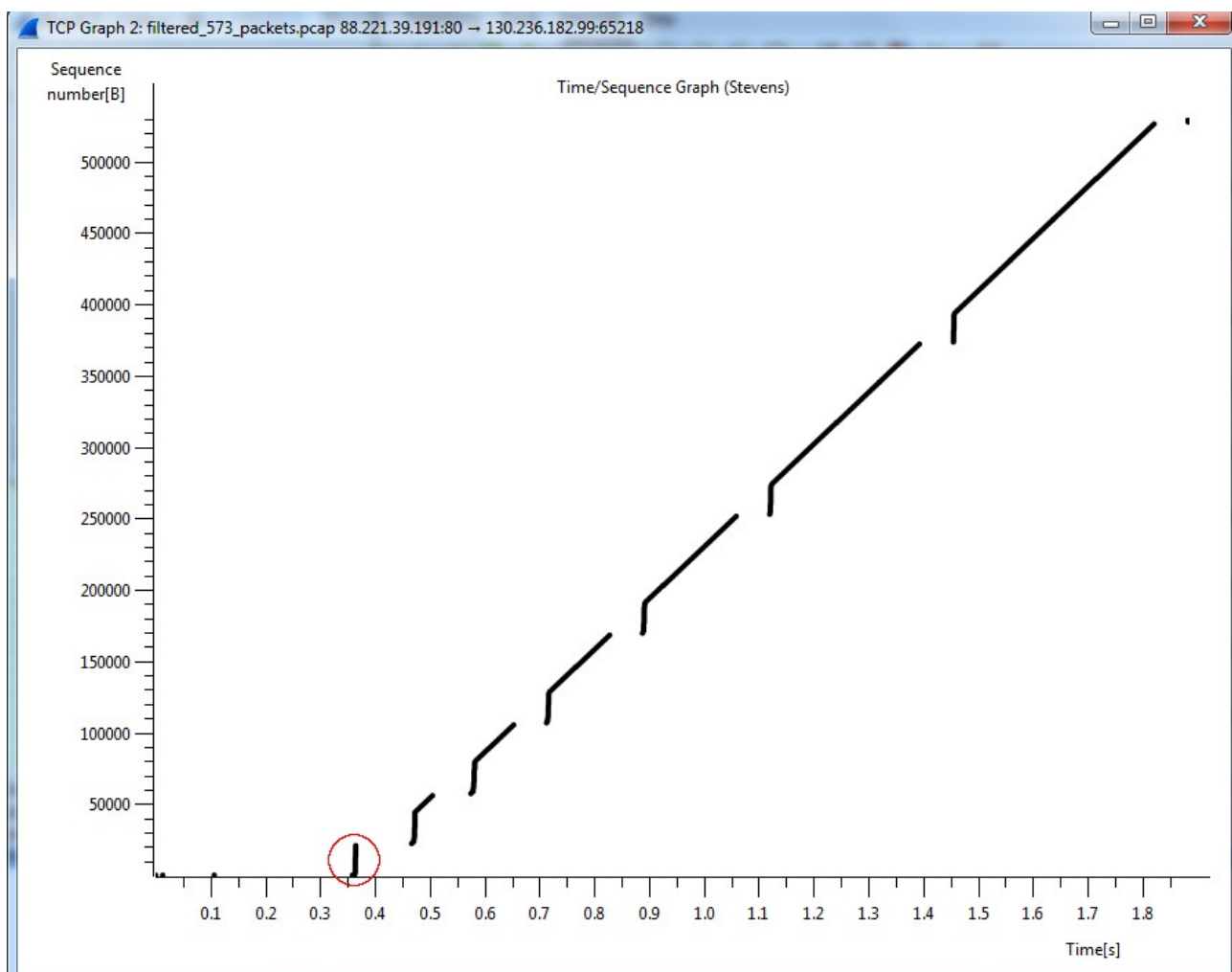


Figure 5.

14. Explain the relationship between (i) the congestion window, (ii) the receiver advertised window, (iii) the number of unacknowledged bytes, and (iv) the effective window at the sender.

Congestion Window:

The window of packets that the sender can send before receiving an ACK. This grows in size exponentially until a timeout occurs, at which point it either resets to 1, is set to half size, or slow start is initiated, depending on what type of protocol you are following.

Receiver advertised window:

This window is compared to the congestion window to determine how much data the sender can send at any time.

Unacknowledged bytes:

The number of bytes whose packets currently haven't been ACK-ed in the active window (that have been sent).

Effective Window:

The effective window is determined by taking the advertised window and removing the total amount of unACK-ed bytes, i.e.:

Effective Window = Advertised Window - (Last Byte Sent - Last Byte ACK-ed).

15. Is it generally possible to find the congestion window size (i.e. *cwnd*) and how it changes with time, from the captured trace files? If so, please explain how. If not, please explain when and when not. Motivate your answer and give examples. Your answer may also benefit from trying to describe and discuss your answer in the context of the two prior questions, for example.

You might be able to see the congestion window by checking if the sender is not sending the maximum size of packets it can send. This is however not 100% accurate, since it can depend on many different things, such as total data size.

Discussion:

There are a lot of different things to keep in mind when talking about TCP, and everything relates to one another very much. When you actually do realize how everything links together though, it becomes quite obvious that it is that way.

4. Task C

16. What is the throughput of each of the connections in bps (bits per second)? What is the total bandwidth of the host on which the clients are running? Discuss the TCP fairness for this case.

Connection 1: $(165095720 * 8) / 521 = 2535059$ bps (~2,54 Mbit/s)

Connection 2: $(165842766 * 8) / 521 = 2546529$ bps (~2,55 Mbit/s)

Connection 3: $(165458792 * 8) / 514 = 2575234$ bps (~2,58 Mbit/s)

Connection 4: $(16325772 * 8) / 512 = 255090$ bps (~0,26 Mbit/s)

Total bandwidth: $2535059 + 2546529 + 2575234 + 255090 = 7911912$ bps (~8 Mbit/s)

Is this fair?

Yes, since the last file is approximately 10 times smaller than the others, and is receiving approximately 10 times less bandwidth.

If you take RTT into consideration, it is still fair, as the RTT is the same for all of the connections.

17. What is the throughput of each of the connections in bps (bits per second)? What is the total bandwidth of the host on which the clients are running? Discuss the TCP fairness for this case.

Connection 1: $(261319130 * 8) / 90 = 23228367$ bps (~23,23 Mbit/s)

Connection 2: $(175995832 * 8) / 90 = 15644073$ bps (~15,64 Mbit/s)

Connection 3: $(151894552 * 8) / 90 = 13501737$ bps (~13,50 Mbit/s)

Connection 4: $(140388568 * 8) / 90 = 12478983$ bps (~12,48 Mbit/s)

Connection 5: $(108610702 * 8) / 90 = 9654284$ bps (~9,65 Mbit/s)

Connection 6: $(70644690 * 8) / 90 = 6279528$ bps (~6,28 Mbit/s)

Connection 7: $(65744938 * 8) / 90 = 5843994$ bps (~5,84 Mbit/s)

Connection 8: $(43212876 * 8) / 90 = 3841144$ bps (~3,84 Mbit/s)

Connection 9: $(39222524 * 8) / 90 = 3486446$ bps (~3,49 Mbit/s)

Total bandwidth: $23228367 + 15644073 + 13501737 + 12478983 + 9654284 + 6279528 + 5843994 + 3841144 + 3486446 = 93958556$ bps (~93 Mbit/s)

Is this fair?

Yes, because the difference in size reflects the difference in speed. In other words, (first packet size / first packet speed) \approx (second packet size / second packet speed).

If you take RTT into consideration, however, the differences in relative speed are very different. Of course, this is to be expected, since if the RTT increases, the host has to wait a larger amount of time before sending the next packet/batch of packets.

18. Discuss the TCP fairness for this case. For all of these questions you must take a closer look at the relationships between the characteristics of the different connections and discuss your findings in the context of the different experiments. You are expected to show that you understand the concept of TCP fairness and how the different scenarios may impact the throughput relationships that you observe and those that you may expect in general. To help the discussion you may for example want to create a scatter plot that show the estimated round trip time (RTT) and throughput against each other (for the different connections). You also want to carefully examine and discuss the above throughput equation and how it may apply to each scenario.

Connection 1: $(108851134 * 8) / 58 = 15013949$ bps (~15,01 Mbit/s)

Connection 2: $(90435681 * 8) / 58 = 12473887$ bps (~12,47 Mbit/s)

Connection 3: $(57971584 * 8) / 53 = 8750427$ bps (~8,75 Mbit/s)

Connection 4: $(32000012 * 8) / 29 = 8827589$ bps (~8,82 Mbit/s)

Connection 5: $(32557334 * 8) / 35 = 7441676$ bps (~7,44 Mbit/s)

Connection 6: $(27099361 * 8) / 31 = 6993383$ bps (~6,99 Mbit/s)

Connection 7: $(26329578 * 8) / 31 = 6794729$ bps (~6,79 Mbit/s)

Connection 8: $(38834490 * 8) / 56 = 5547784$ bps (~5,55 Mbit/s)

Connection 9: $(23571761 * 8) / 35 = 5387831$ bps (~5,39 Mbit/s)

Connection 10: $(36252962 * 8) / 55 = 5273158$ bps (~5,27 Mbit/s)

Is this fair?

It is difficult to determine whether this is fair or not, as the data presented is for 10 different people. There are a lot of variables that could determine whether it is fair or not, such as different internet

speeds. If we however assume that all the conditions are the same, the speeds are not fair, seeing as the speed is not relative to the size of the file/packet downloaded.

If you also take RTT into consideration, it is still unfair, since even when taking that into consideration, the differences in relativity are very different.

Discussion:

Different connection types require largely different thinking patterns about TCP fairness, since you have to take RTT and other things into consideration when determining whether or not it is, in fact, fair.