IK2218 Lab 1 report

Felix Hedenström
Robin Gunning
Mikael Forsberg

September 24, 2017

Contents

10 Basic operation of TCP **2**
10.7 2

11 TCP connection management **2**
11.1.6 2
11.1.7 3
11.2.5 3

12 TCP data transfer **3**
12.1.4 3
12.1.5 4
12.2.5 4

13 TCP retransmissions **4**
13.6 4

14 TCP congestion control **4**
14.6 4

Acknowledgement: Host B did not use the reno TCP congestion avoidance algorithm until question 13. With our knowledge of the assignment this should not matter, but it best to disclose everything.

10 Basic operation of TCP

10.7

a) Through wireshark we counted 15 packets in total

b) We found these sequence numbers in the packages sent. They are easiest presented as relative sequence numbers: 0, 1, 1001, 2449, 5345, 6793, 10002

   The range of the numbers should then be described as [0-10002]

c) 10 packets did not carry any payload

d) The packages carried: 11 006 bytes total of which 10 000 bytes where user data

e) UDP is more efficient in terms of overhead

   In this case:
   UDP has an efficiency of 95.97%
   TCP has an efficiency of 90.86% = 10 000 / 11 006

11 TCP connection management

11.1.6

a) The first three packets are the three-way handshake. The packets were:

   First package was from A to B, containing the flag SYN.
   The second package was from B to A, containing the flags SYN and ACK.
   The third package went from A to B and contained only the flag ACK.

   This means that host A wants to communicate with server B through a certain socket, B answers by acknowledging this and opening the socket, A answers that it has acknowledged that the socket is open. SYN means they should synchronize the sequence numbers, so that the relative sequence numbers are the same.

b) Both the host and the client had the same initial relative sequence number, 0. This is in line with the results we observed in b).

   Host A, the client, had an absolute initial sequence number of 2 368 580 297
   Host B, the server, had an absolute initial sequence number of 2 235 071 071

c) The fourth package was the first application data package.
   It was a telnet data package and carried 27 bytes of data.

d) The client/host A had an initial window size of 29200
The server/host B had an initial windows size of 28960

e) Roughly 1.8 milliseconds was observed through Wireshark

11.1.7

a) The last three packets are involved in the connection termination. This is basically a handshake terminating the connection.

b) The first package was from A to B and contained the flags FIN and ACK. The second package was from B to A and contained the flags FIN and ACK.
The last package was from A to B and contained a final ACK.

So the client wants to end the connections and sends the FIN bit, indication that the host has finished the transmission and wants to end the connection. The host also sends a FIN bit and an ACK to show that it has seen the hosts FIN packet. The host then sends a final ACK to respond that it has seen that the server also is ready to finish/terminate the connection.

11.2.5

a) The server replies to the client's initial SYN with an RST/ACK . The RST flag means that the host should stop using the TCP connection as soon as possible.

b) The connection is terminated in less than one millisecond.

12 TCP data transfer

12.1.4

a) Three segments per typed character can be seen.

b) The payloads consists of the single typed character. This is true for both the client and the server. The first segment contains the character payload and is sent from the client to the server. The second segment contains the same character payload and is sent from the server to the client. The third segment does not contain any payload.

c) This is because the server uses piggybacking and sends an ACK with the character data it sends back.

The process of sending a single character is:

Host A sends a character Host B sends back the same character and acknowledges that A has sent a character Host A acknowledges that B sent back a character.

d) The delay is because of the delayed ack algorithm. By observing Wireshark the delay can be estimated to be approximately 150 milliseconds.

e) The window size of the client is 245.
The window size of the server is 227.

The windows sizes of the client and server did not vary during the connection.

12.1.5

a) When typing quickly enough, multiple characters are sent in single packets. The client seemingly only needs to send one ACK back after receiving many rapid mirrored characters back from the server. This is because the delayed ack algorithm.

12.2.5

a) The rule (according to one of our TA:s) is that the receiver waits for some time to accumulate received data and then ACKs it. This is according to the delayed ack algorithm. It waits up to 500 ms by default. In our data it is really difficult to make out any type of pattern that would support this, and we do not see this rule.

b) It scales with the window size. However when reading the data the receiver usually acknowledge 1448 bytes of data. There are very few examples where the ack is larger.

c) The window size increases

d) 0.030413273 to 0.033449434. So roughly about 0.003 seconds.

e) No. There tends to be quite alot of headroom.

13 TCP retransmissions

13.6

a) There is only one packet sent at retransmission timeout.

b) The retransmissions end when the connection is closed due to session timeout.

14 TCP congestion control

14.6

a) Yes, it does. The window starts out small and increases in size as time and packets passes.

b) Fast recovery (in the congestion avoidance phase) would result in a lowering of the CWND, which with the artificial delay enabled should be visible in the tcptrace graph in the form of a reduction in the number of segments sent per burst. However, it seems that in our test we only had packet losses during the slow start phase.