Lab5.Transport.Layer

Jack Malone

November 2021

1 Question 1

Source Port: 443	Destination Port: 51421
Sequence number: 1317 (relative sequence number)	
acknowledgement number: 1683 (relative sequence number)	
data offset: NA	reserved: not set
control flags: 000000010000	window size: 8122
checksum: 0x432b	urgent pointer:0
optional data	

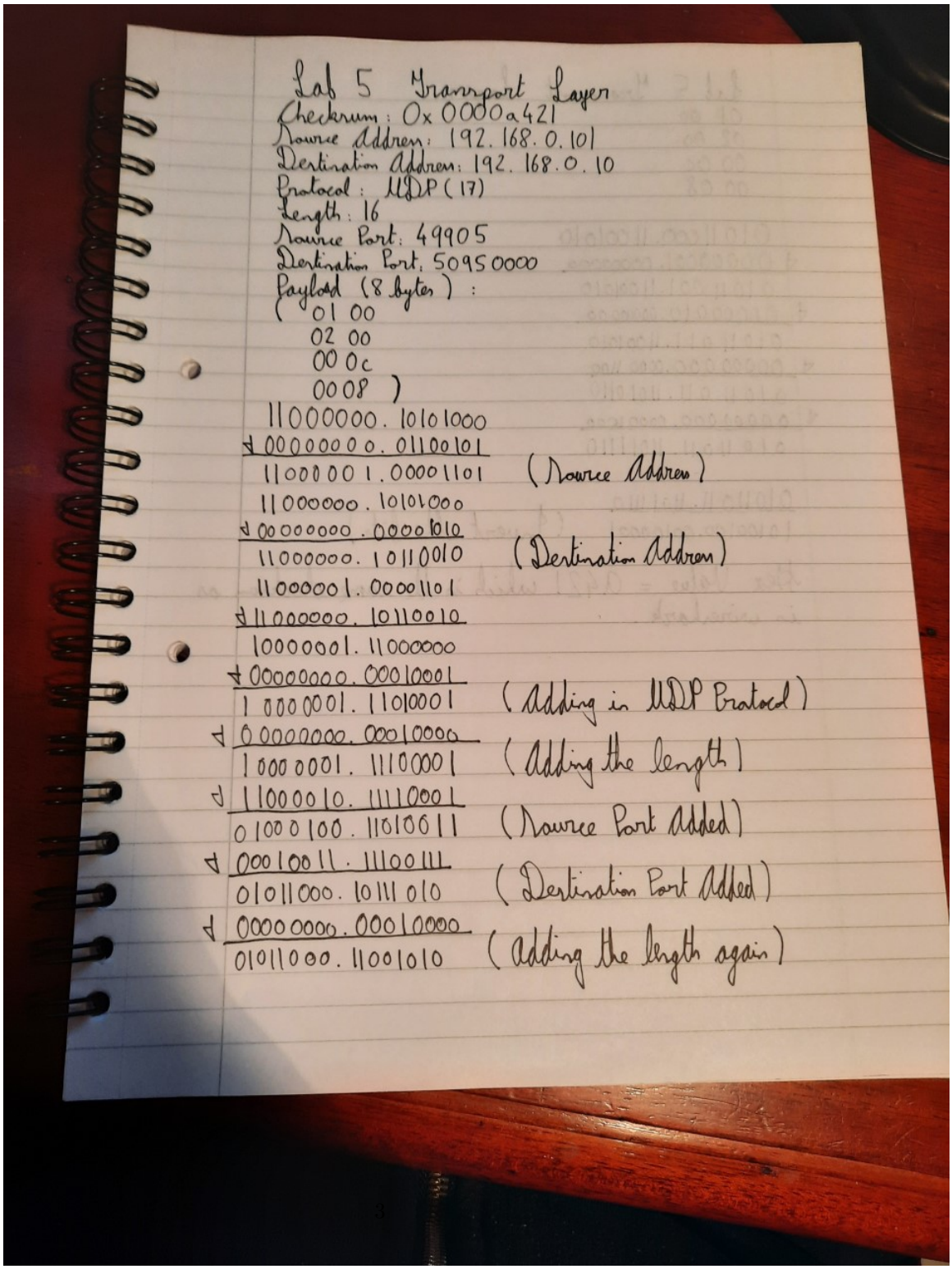
- source port: The TCP Source Port is the port number used by the computer sending the TCP segment and is usually a number above 1024.
- destination port: the TCP destination port is the port number used by the computer receiving the TCP packet and is usually a number below 1024.
- sequence number: The sequence number helps the TCP software on both sides keep track of how much data has been transferred and to put the data back into the correct order if it is received in the wrong order, and to request data when it has been lost in transit.
- data offset: The TCP data offset indicates the number of bytes into the TCP packet where data can be found.
- window: Number of octets in the TCP header.
- checksum: A cyclic redundancy check checksum is calculated by the sender and added to this field before transmission. This field is used by the receiver to verify the integrity of the data in the TCP payload and reject data that fails the CRC check.
- urgent pointer: Point to the end of "urgent" data in the packet, but this field only exists if the URG flag is set.
- data: This field contains a segment of data from the user application, such as part of an email or web page.

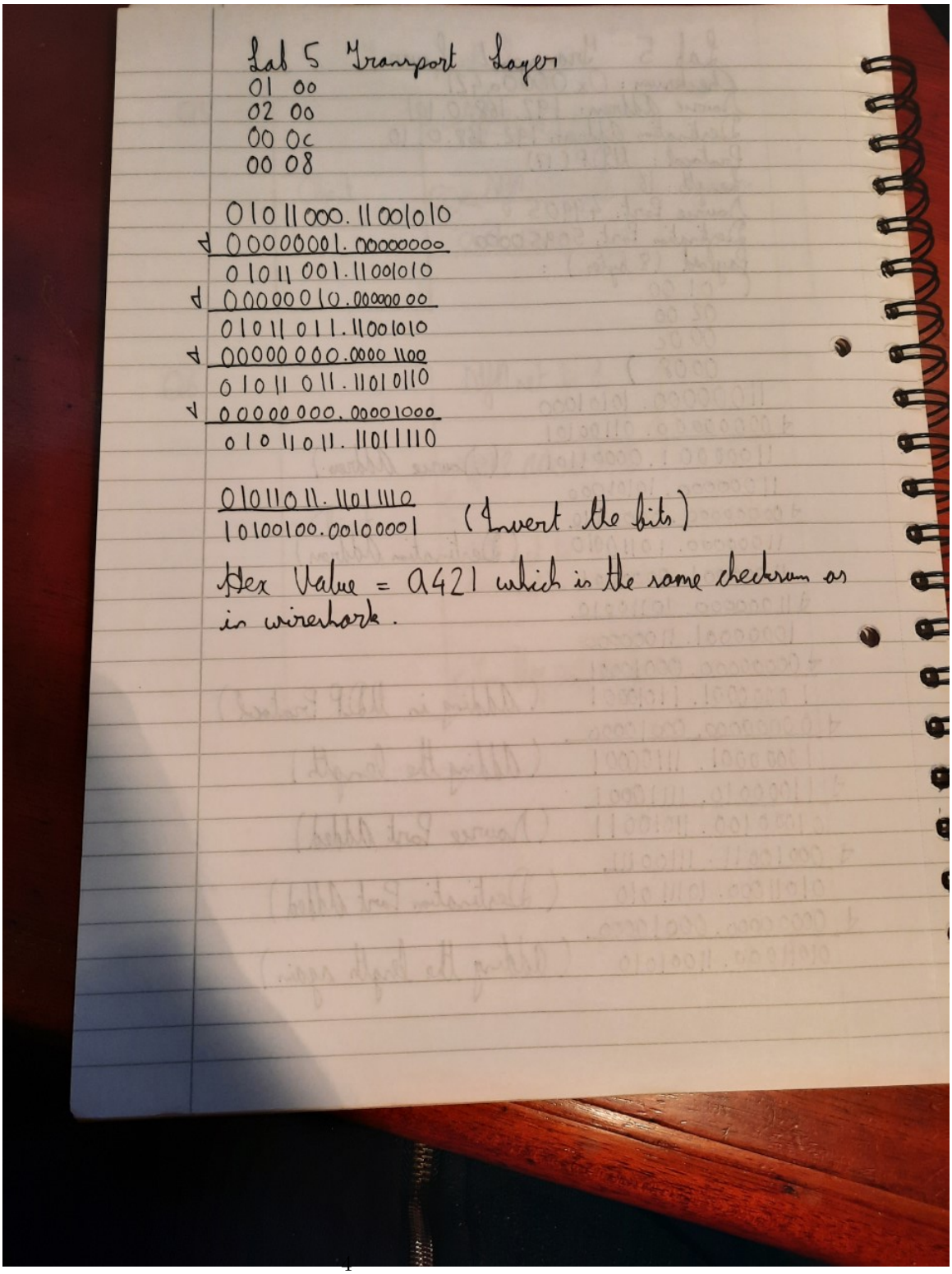
2 Question 2

Source Port: 443	Destination Port: 54186
UDP length: 254	Checksum: 0x2f7a

- source port: the port of the device sending the data. This field can be set to 0 if the destination computer doesn't need to reply to the sender.
- destination port: the port of the device receiving the data. UDP port numbers can be between 0 and 65,535.
- length: Specifies the number of bytes comprising the UDP header and the UDP payload data.
- checksum: the checksum allows the receiving device to verify the integrity of the packet header and payload.

3 Question 3


Lab 5 Transport Layer
Checksum: 0x0000a421
Source Address: 192.168.0.101
Destination Address: 192.168.0.10
Protocol: UDP (17)
Length: 16
Source Port: 49905
Destination Port: 50950000
Payload (8 bytes):
01 00
02 00
00 0c
0008
11000000.10101000
+00000000.01100101
11000001.00001101 (Source Address)
11000000.10101000
+00000000.00001010
11000000.10110010 (Destination Address)
11000001.00001101
+11000000.10110010
10000001.11000000
+00000000.00010001
10000001.11010001 (Adding in UDP Protocol)
+00000000.00010000
10000001.11100001 (Adding the length)
+11000010.11110001
01000100.11010011 (Source Port Added)
+00010011.11100111
01011000.10111010 (Destination Port Added)
+00000000.00010000
01011000.11001010 (Adding the length again)



Lab 5 Transport Layer

01 00
02 00
00 0c
00 08

0101 1000.1100 1010
+ 0000 0001.0000 0000
0101 1001.1100 1010
+ 0000 0010.0000 0000
0101 1011.1100 1010
+ 0000 0000.0000 1100
0101 1011.1101 0110
+ 0000 0000.0000 1000
0101 1011.1101 1110

0101 1011.1101 1110
1010 0100.0010 0001 (Invert the bits)

Hex Value = A421 which is the same checksum as in wireshark.

4 Question 4

When I opened up a live video on youtube, the main protocol that youtube was using was UDP because UDP is much faster than TCP as it does not need to do the same amount of error checking as TCP. This is useful for live videos as it has to get the data that the person is livestreaming to the platform as quickly as possible to the people watching so that they don't have any lag in the video and so that they can watch it in as good of a video quality as possible.
When I switched to watching a non live video that was already on the youtube servers, it switched to using TCP as it did not have to get all of the information about the video from the person livestreaming. This meant that they could use the time saved from not having to fetch the video from the streamer to do error checking on the data being sent to the person watching the video as well as making sure that all of the packets make it to the viewer so there isn't any data loss while they are watching the video.
As a result of the different protocols that live videos and pre-recorded videos have on the platform, I noticed that in general the pre-recorded videos are usually better quality for me as they are using a protocol that ensures that all of the data gets to the end user. For livestreams the video usually ends up as lower quality as it doesn't have time to send all of the video information and I will usually need to watch in a slightly lower video quality than normal.

5 Question 5

The tcp 3-way handshake is a process which is used in a TCP/IP network to make a connection between the server and client. It is a three-step process that requires both the client and server to exchange synchonisation acknowledgment packets before the real data communication process starts.
Diagrams for this question and question 6 are located at the end of the document.
The handshake process is designed in such a way that both ends help you to initiate, negotiate and separate TCP socket connections at the same time.

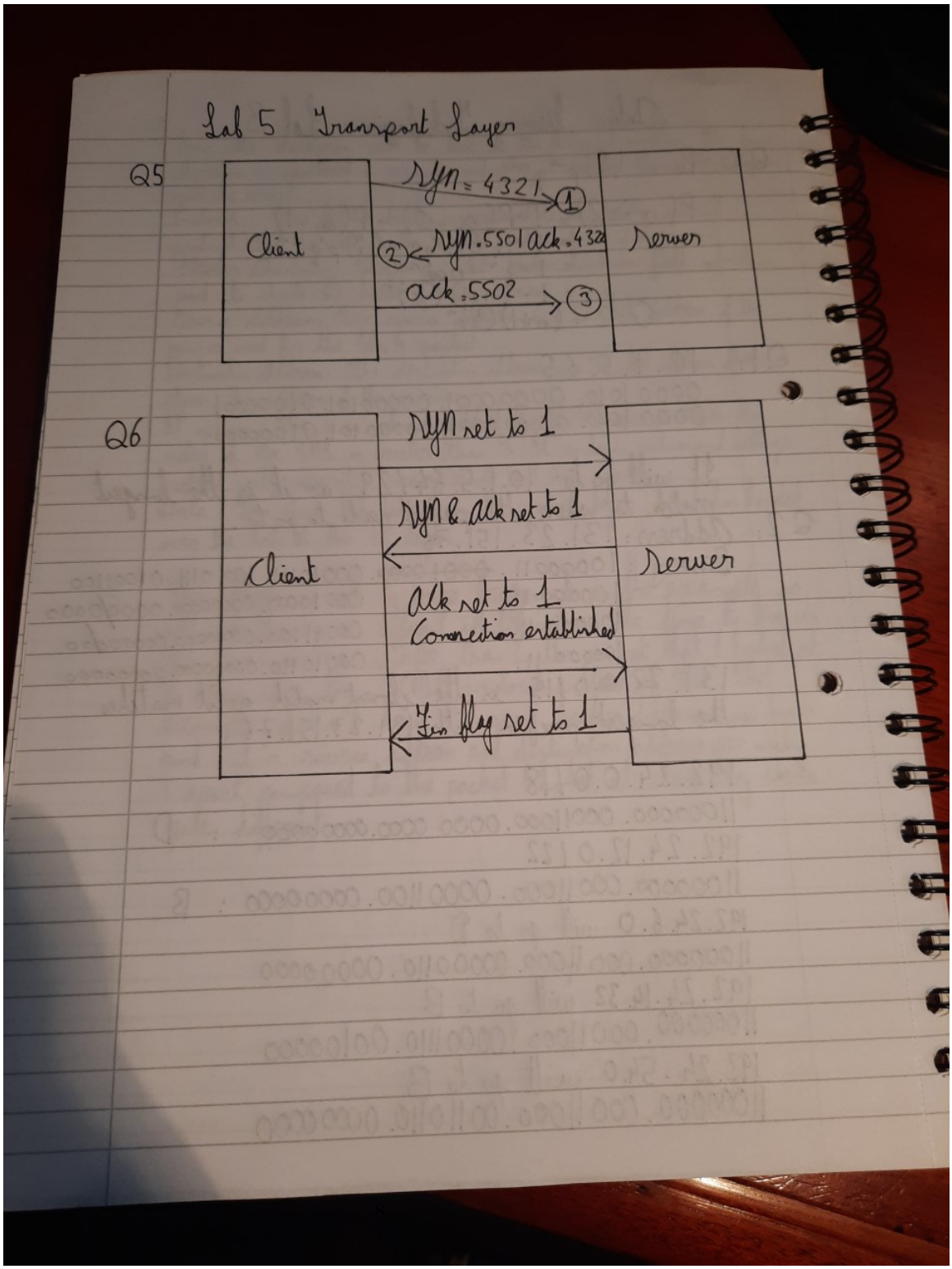
- Step 1: In the first step, the client establishes a connection with a server. It sends a segment with SYN and informs the server about the client should start communication, and with what should be its sequence number.
- Step 2: In this step the server responds to the client request with SYN-Ack signal set. ACK helps to signify the response of the segment that is received and SYN signifies what sequence number it should be able to start with the segments.
- Step 3: In the final step, the client acknowledges the response of the server, and they both create a stable connection that will begin the actual data transfer process.

6 Question 6

A TCP-4 way teardown is initiated after a TCP 3-way handshake occurs and the server wants to finish the connection to the client. The connection is terminated after the server sends the client the FIN flag.
There are four steps to the process.

- Step 1: send syn from the client to the server.
- Step 2: send syn/ack from the server to the client.
- Step 3: send ack from the client to the server.
- Step 4: the server sets the fin flag and ends the connection.

7 Diagrams


Lab 5 Transport Layer
Q5
Client
Syn = 4321
1
2
Syn.5501 ack.4322
Server
ack.5502
3
Q6
Client
Syn set to 1
Syn & ack set to 1
ack set to 1
Connection established
Fin flag set to 1
Server