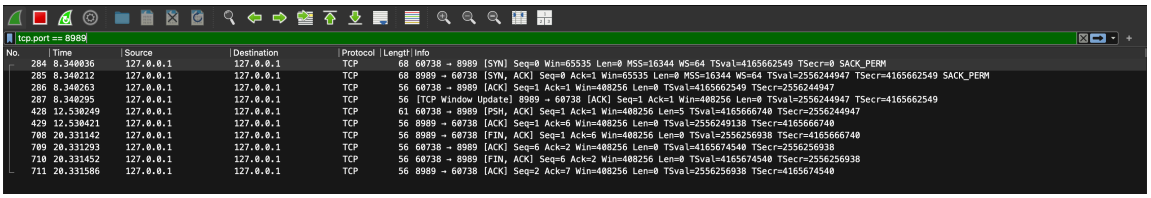
Leandro Lika
Lucca Alt
Nillan Sivarasa

Task 1 TCP Traffic

a)



The first three rows describes the TCP-Handshake. The client sends a synchronize packet (SYN) to the server to start the connection. In the figure this is packet number 284, where the source port is 60738 and the destination port is 8989 (server). This packet includes a sequence number (Seq=0) which is used to synchronize the session. In the second row the server responds with a [SYN,ACK] packet, acknowledging the received SYN (by setting ACK=1) and sending its own SYN. In the third row we can observe that the client sends an ACK packet back to the server, finishing the handshake. This is packet No. 286, where ACK=1 confirms the Servers SYN.

In the fifth row we can observe the occurrence of data transfer after sending a message in our case "Hey" from the client to the server. Packet No. 428 shows a PSH (push) and ACK flags set. PSH tells the receiver to pass the data to the application and ACK acknowledges the previous packets. Packet No. 429 shows that the server acknowledges the send packet.

Finally, we can observe the packets involved in closing the connections. The Server sends a FIN (finish) and ACK packet to the client, signaling that no further data will be sent and the connection should be closed. The sequence value Seq=1 and the acknowledge value ACK=6 are set. This means that this packet indicates the last byte stream from the server and at the same time confirms all data received up to this point. The Client confirms the receipt of the FIN packet by sending an ACK. The confirmation message ACK=2 confirms receipt of the final FIN packet from the server. After acknowledging the FIN the client sends one itself and an ACK packet to close the connection from his side aswell. The Sequence value (Seq=6) indicates the continuation od the sequencing and the acknowledgement value ACK=2 continues to acknoweledge the received packets. Finally The server receives the FIN packet from the client and sends a final ACK packet in response. This confirms receipt of the FIN packet from the client and completes the four-way handshake.

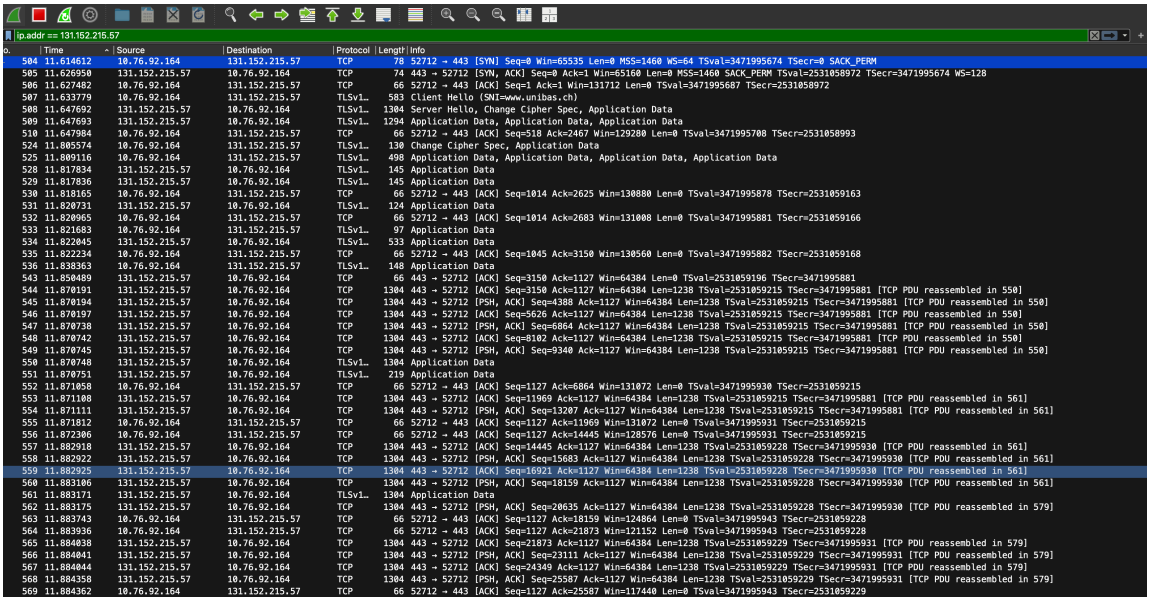
b)

```python
def start_client(ip, port):

    # Create a TCP/IP socket
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #(AF_INET) indicates that we are using an internet socket /
                                                                      #The second parameter stands for the protocol we want to use. SOCK_STREAM indicates that we are using TCP and not UDP.

    # Connect the socket to the server's address and port
    client_socket.connect((ip, port))

    # Create a separate thread for receiving messages
    receive_thread = threading.Thread(target=receive_messages, args=(client_socket,)) #The receive_messages function takes the client socket as an argument to read data from it.
    receive_thread.start()

     # Continuously send messages
    while True:
        message = input()
        if message == '':
            break
        client_socket.send(message.encode('utf-8')) #Encoding in UTF-8 is necessary as network communication requires data to be in bytes rather than in string format.
    # Close the connection and the client socket
    print("Closing connection...")
    client_socket.close()

    pass
```

Task 4 - HTTPS Traffic

Start recording network traffic with Wireshark and open any https website (we chose https://www.unibas.ch/de). What is a big difference in the content of the relevant packets, especially with respect to Task 1?

Besides the the TCP traffic seen in the first task, the image shows network traffic from unibas.ch that is encrypted using TLS (Transport Layer Security). TLS secures the transmission of data by encrypting the data itself as well as verifying the identity of the communication partners. It shows how data is sent and received via the port 443 (HTTPS). The packets 507 and 508 shows the TLS-Handshake with "Client Hello" and "Server Hello".



ip.addr == 131.152.215.57

o.	Time	Source	Destination	Protocol	Length	Info
504	11.614612	10.76.92.164	131.152.215.57	TCP	78	52712 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=3471995674 TSecr=0 SACK_PERM
505	11.626950	131.152.215.57	10.76.92.164	TCP	74	443 → 52712 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2531058972 TSecr=3471995674 WS=128
506	11.627482	10.76.92.164	131.152.215.57	TCP	66	52712 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=3471995687 TSecr=2531058972
507	11.633779	10.76.92.164	131.152.215.57	TLSv1…	583	Client Hello (SNI=www.unibas.ch)
508	11.647692	131.152.215.57	10.76.92.164	TLSv1…	1304	Server Hello, Change Cipher Spec, Application Data
509	11.647693	131.152.215.57	10.76.92.164	TLSv1…	1294	Application Data, Application Data, Application Data
510	11.647984	10.76.92.164	131.152.215.57	TCP	66	52712 → 443 [ACK] Seq=518 Ack=2467 Win=129280 Len=0 TSval=3471995708 TSecr=2531058993
524	11.805574	10.76.92.164	131.152.215.57	TLSv1…	130	Change Cipher Spec, Application Data
525	11.809116	10.76.92.164	131.152.215.57	TLSv1…	498	Application Data, Application Data, Application Data, Application Data
528	11.817834	131.152.215.57	10.76.92.164	TLSv1…	145	Application Data
529	11.817836	131.152.215.57	10.76.92.164	TLSv1…	145	Application Data
530	11.818165	10.76.92.164	131.152.215.57	TCP	66	52712 → 443 [ACK] Seq=1014 Ack=2625 Win=130880 Len=0 TSval=3471995878 TSecr=2531059163
531	11.820731	131.152.215.57	10.76.92.164	TLSv1…	124	Application Data
532	11.820965	10.76.92.164	131.152.215.57	TCP	66	52712 → 443 [ACK] Seq=1014 Ack=2683 Win=131008 Len=0 TSval=3471995881 TSecr=2531059166
533	11.821683	10.76.92.164	131.152.215.57	TLSv1…	97	Application Data
534	11.822045	131.152.215.57	10.76.92.164	TLSv1…	533	Application Data
535	11.822234	10.76.92.164	131.152.215.57	TCP	66	52712 → 443 [ACK] Seq=1045 Ack=3150 Win=130560 Len=0 TSval=3471995882 TSecr=2531059168
536	11.838363	10.76.92.164	131.152.215.57	TLSv1…	148	Application Data
543	11.850489	131.152.215.57	10.76.92.164	TCP	66	443 → 52712 [ACK] Seq=3150 Ack=1127 Win=64384 Len=0 TSval=2531059196 TSecr=3471995881
544	11.870191	131.152.215.57	10.76.92.164	TCP	1304	443 → 52712 [ACK] Seq=3150 Ack=1127 Win=64384 Len=1238 TSval=2531059215 TSecr=3471995881 [TCP PDU reassembled in 550]
545	11.870194	131.152.215.57	10.76.92.164	TCP	1304	443 → 52712 [PSH, ACK] Seq=4388 Ack=1127 Win=64384 Len=1238 TSval=2531059215 TSecr=3471995881 [TCP PDU reassembled in 550]
546	11.870197	131.152.215.57	10.76.92.164	TCP	1304	443 → 52712 [ACK] Seq=5626 Ack=1127 Win=64384 Len=1238 TSval=2531059215 TSecr=3471995881 [TCP PDU reassembled in 550]
547	11.870738	131.152.215.57	10.76.92.164	TCP	1304	443 → 52712 [PSH, ACK] Seq=6864 Ack=1127 Win=64384 Len=1238 TSval=2531059215 TSecr=3471995881 [TCP PDU reassembled in 550]
548	11.870742	131.152.215.57	10.76.92.164	TCP	1304	443 → 52712 [ACK] Seq=8102 Ack=1127 Win=64384 Len=1238 TSval=2531059215 TSecr=3471995881 [TCP PDU reassembled in 550]
549	11.870745	131.152.215.57	10.76.92.164	TCP	1304	443 → 52712 [PSH, ACK] Seq=9340 Ack=1127 Win=64384 Len=1238 TSval=2531059215 TSecr=3471995881 [TCP PDU reassembled in 550]
550	11.870748	131.152.215.57	10.76.92.164	TLSv1…	1304	Application Data
551	11.870751	131.152.215.57	10.76.92.164	TLSv1…	219	Application Data
552	11.871058	10.76.92.164	131.152.215.57	TCP	66	52712 → 443 [ACK] Seq=1127 Ack=6864 Win=131072 Len=0 TSval=3471995930 TSecr=2531059215
553	11.871108	131.152.215.57	10.76.92.164	TCP	1304	443 → 52712 [ACK] Seq=11969 Ack=1127 Win=64384 Len=1238 TSval=2531059215 TSecr=3471995881 [TCP PDU reassembled in 561]
554	11.871111	131.152.215.57	10.76.92.164	TCP	1304	443 → 52712 [PSH, ACK] Seq=13207 Ack=1127 Win=64384 Len=1238 TSval=2531059215 TSecr=3471995881 [TCP PDU reassembled in 561]
555	11.871812	10.76.92.164	131.152.215.57	TCP	66	52712 → 443 [ACK] Seq=1127 Ack=11969 Win=131072 Len=0 TSval=3471995931 TSecr=2531059215
556	11.872306	10.76.92.164	131.152.215.57	TCP	66	52712 → 443 [ACK] Seq=1127 Ack=14445 Win=128576 Len=0 TSval=3471995931 TSecr=2531059215
557	11.882918	131.152.215.57	10.76.92.164	TCP	1304	443 → 52712 [ACK] Seq=14445 Ack=1127 Win=64384 Len=1238 TSval=2531059228 TSecr=3471995930 [TCP PDU reassembled in 561]
558	11.882922	131.152.215.57	10.76.92.164	TCP	1304	443 → 52712 [PSH, ACK] Seq=15683 Ack=1127 Win=64384 Len=1238 TSval=2531059228 TSecr=3471995930 [TCP PDU reassembled in 561]
559	11.882925	131.152.215.57	10.76.92.164	TCP	1304	443 → 52712 [ACK] Seq=16921 Ack=1127 Win=64384 Len=1238 TSval=2531059228 TSecr=3471995930 [TCP PDU reassembled in 561]
560	11.883106	131.152.215.57	10.76.92.164	TCP	1304	443 → 52712 [PSH, ACK] Seq=18159 Ack=1127 Win=64384 Len=1238 TSval=2531059228 TSecr=3471995930 [TCP PDU reassembled in 561]
561	11.883171	131.152.215.57	10.76.92.164	TLSv1…	1304	Application Data
562	11.883175	131.152.215.57	10.76.92.164	TCP	1304	443 → 52712 [PSH, ACK] Seq=20635 Ack=1127 Win=64384 Len=1238 TSval=2531059228 TSecr=3471995930 [TCP PDU reassembled in 579]
563	11.883736	10.76.92.164	131.152.215.57	TCP	66	52712 → 443 [ACK] Seq=1127 Ack=18159 Win=124864 Len=0 TSval=3471995943 TSecr=2531059228
564	11.883936	10.76.92.164	131.152.215.57	TCP	66	52712 → 443 [ACK] Seq=1127 Ack=21873 Win=121152 Len=0 TSval=3471995943 TSecr=2531059228
565	11.884038	131.152.215.57	10.76.92.164	TCP	1304	443 → 52712 [ACK] Seq=21873 Ack=1127 Win=64384 Len=1238 TSval=2531059229 TSecr=3471995931 [TCP PDU reassembled in 579]
566	11.884041	131.152.215.57	10.76.92.164	TCP	1304	443 → 52712 [PSH, ACK] Seq=23111 Ack=1127 Win=64384 Len=1238 TSval=2531059229 TSecr=3471995931 [TCP PDU reassembled in 579]
567	11.884044	131.152.215.57	10.76.92.164	TCP	1304	443 → 52712 [ACK] Seq=24349 Ack=1127 Win=64384 Len=1238 TSval=2531059229 TSecr=3471995931 [TCP PDU reassembled in 579]
568	11.884358	131.152.215.57	10.76.92.164	TCP	1304	443 → 52712 [PSH, ACK] Seq=25587 Ack=1127 Win=64384 Len=1238 TSval=2531059229 TSecr=3471995931 [TCP PDU reassembled in 579]
569	11.884362	10.76.92.164	131.152.215.57	TCP	66	52712 → 443 [ACK] Seq=1127 Ack=25587 Win=117440 Len=0 TSval=3471995943 TSecr=2531059229