

Andre Hei Wang Law

40175600

Design Document - Wireshark

Task:

Open the Wireshark while running the client-server program, and capture the packet that is sent from server to the client. To recognize the packets that are being transferred from the server program, check the source and destination addresses. The source and destination address will be 127.0.0.1 as you are using localhost

For TCP:

- What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client and server? What is in the segment that identifies as a SYN segment?

The image shows a Wireshark packet capture of a TCP connection. The packet list at the top shows several packets, with packet 442 highlighted. The packet details pane shows the structure of the packet, including the Ethernet II header, Internet Protocol Version 4 header, and Transmission Control Protocol header. The TCP header shows the source port as 51195, destination port as 9990, and the sequence number as 0. The flags field shows the SYN flag set.

Packet list:

No.	Time	Source	Destination	Protocol	Length	Info
442	27.829106	127.0.0.1	127.0.0.1	TCP	56	51195 → 9990 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
443	27.829143	127.0.0.1	127.0.0.1	TCP	56	9990 → 51195 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 S
444	27.829160	127.0.0.1	127.0.0.1	TCP	44	51195 → 9990 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
445	29.463120	127.0.0.1	127.0.0.1	TCP	56	51196 → 49350 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
446	29.463150	127.0.0.1	127.0.0.1	TCP	56	49350 → 51196 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256

Packet details:

Frame 442: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_{...}, id 0

Null/Loopback

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

Transmission Control Protocol, Src Port: 51195, Dst Port: 9990, Seq: 0, Len: 0

Source Port: 51195

Destination Port: 9990

[Stream index: 48]

[Conversation completeness: Incomplete, DATA (15)]

[TCP Segment Len: 0]

Sequence Number: 0 (relative sequence number)

Sequence Number (raw): 1345079190

[Next Sequence Number: 1 (relative sequence number)]

Acknowledgment Number: 0

Acknowledgment number (raw): 0

1000 ... Header Length: 32 bytes (8)

Flags: 0x002 (SYN)

000. ... = Reserved: Not set

...0 ... = Accurate ECN: Not set

...0 ... = Congestion Window Reduced: Not set

...0 ... = ECN-Echo: Not set

...0 ... = Urgent: Not set

...0 ... = Acknowledgment: Not set

...0 ... = Push: Not set

...0 ... = Reset: Not set

...1 ... = Syn: Set

...0 ... = Fin: Not set

[TCP Flags:S.]

Window: 65535

[Calculated window size: 65535]

Checksum: 0xf025 [unverified]

[Checksum Status: Unverified]

Urgent Pointer: 0

> Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP)

> [Timestamps]

```

Terminal: Local (2) × Local × + ▾
Protocol: tcp, Port: 9990, Debug Mode: ON, ServerIP: 127.0.0.1

Waiting for connections...
TCP Information ('127.0.0.1', 51195)

```

Sequence number of SYN is 0. "Syn" flag is "Set" means that it is a SYN segment.

- What are the sequence numbers of the first two segments in the TCP connection? 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 acknowledgment was received, what is the RTT value for each of the two segments? Build the round-trip time graph.

No.	Time	Source	Destination	Protocol	Length	Info
442	27.829106	127.0.0.1	127.0.0.1	TCP	56	51195 → 9990 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
443	27.829143	127.0.0.1	127.0.0.1	TCP	56	9990 → 51195 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
444	27.829160	127.0.0.1	127.0.0.1	TCP	44	51195 → 9990 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
445	29.463126	127.0.0.1	127.0.0.1	TCP	56	51195 → 49350 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
446	29.463150	127.0.0.1	127.0.0.1	TCP	56	49350 → 51195 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
447	29.463166	127.0.0.1	127.0.0.1	TCP	44	51195 → 49350 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
448	29.463192	127.0.0.1	127.0.0.1	TCP	41132	51195 → 49350 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=41088
449	29.463217	127.0.0.1	127.0.0.1	TCP	44	49350 → 51195 [ACK] Seq=1 Ack=41089 Win=2619648 Len=0
450	29.463229	127.0.0.1	127.0.0.1	TCP	44	51195 → 49350 [FIN, ACK] Seq=41089 Ack=1 Win=2619648 Len=0
451	29.463233	127.0.0.1	127.0.0.1	TCP	44	49350 → 51195 [ACK] Seq=1 Ack=41090 Win=2619648 Len=0
452	29.478827	127.0.0.1	127.0.0.1	TCP	44	49350 → 51195 [FIN, ACK] Seq=1 Ack=41090 Win=2619648 Len=0
453	29.478839	127.0.0.1	127.0.0.1	TCP	44	51195 → 49350 [ACK] Seq=41090 Ack=2 Win=2619648 Len=0
454	29.579331	127.0.0.1	127.0.0.1	TCP	45	51195 → 9990 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=1
455	29.579345	127.0.0.1	127.0.0.1	TCP	44	9990 → 51195 [ACK] Seq=1 Ack=2 Win=2619648 Len=0
456	29.579390	127.0.0.1	127.0.0.1	TCP	76	9990 → 51195 [PSH, ACK] Seq=1 Ack=2 Win=2619648 Len=32
457	29.579401	127.0.0.1	127.0.0.1	TCP	44	51195 → 9990 [ACK] Seq=2 Ack=33 Win=2619648 Len=0

Sequence number of the first two segments in the TCP connection is 0.

First segment sent at time 27.829106. ACK received at time 27.829143. 3rd handshake at time 27.829160.

Second segment sent at time 29.463126. ACK received at time 29.463233. 3rd handshake at 29.579401.

▼ [SEQ/ACK analysis]

[This is an ACK to the segment in frame: 442]

[The RTT to ACK the segment was: 0.000037000 seconds]

[iRTT: 0.000054000 seconds]

RTT for first segment: 0.000054

RTT for second segment: 0.00007

- What is the length of each of the first six TCP segments?

>	TCP payload (41088 bytes)
>	Data (41088 bytes)
>	TCP payload (1 byte)
>	Data (1 byte)

TCP payload (32 bytes)
> Data (32 bytes)

TCP payload (1 byte)
> Data (1 byte)

TCP payload (32 bytes)
> Data (32 bytes)

TCP payload (1 byte)
> Data (1 byte)

[SEQ/ACK analysis]
TCP payload (32 bytes)
> Data (32 bytes)

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

127.0.0.1	127.0.0.1	TCP	44 9990 → 51195 [ACK] Seq=1 Ack=2 Win=2619648 Len=0
127.0.0.1	127.0.0.1	TCP	76 9990 → 51195 [PSH, ACK] Seq=1 Ack=2 Win=2619648 Len=32
127.0.0.1	127.0.0.1	TCP	44 51195 → 9990 [ACK] Seq=2 Ack=36 Win=2619648 Len=0

Since “Win” are all 2619648, the minimum amount of buffer space is 2619648. Thus, it does not throttle.

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

For this connection, there are no retransmitted segments in the trace file. This is due to the fact that no same sequence number appears at two different times, thus no re-requests of previous segments.

- How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing in every other received segment.

Transmission Control Protocol, Src Port: 51195, Dst Port: 9990, Seq: 1, Ack: 1, Len: 1
Source Port: 51195
Destination Port: 9990
[Stream index: 48]
[Conversation completeness: Incomplete, DATA (15)]
[TCP Segment Len: 1]
Sequence Number: 1 (relative sequence number)
Sequence Number (raw): 1345079191
[Next Sequence Number: 2 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 503166755
0101 = Header Length: 20 bytes (5)

The ACK number is 1. I couldn't find in my cases of a receiver ACKing every other received segment.

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

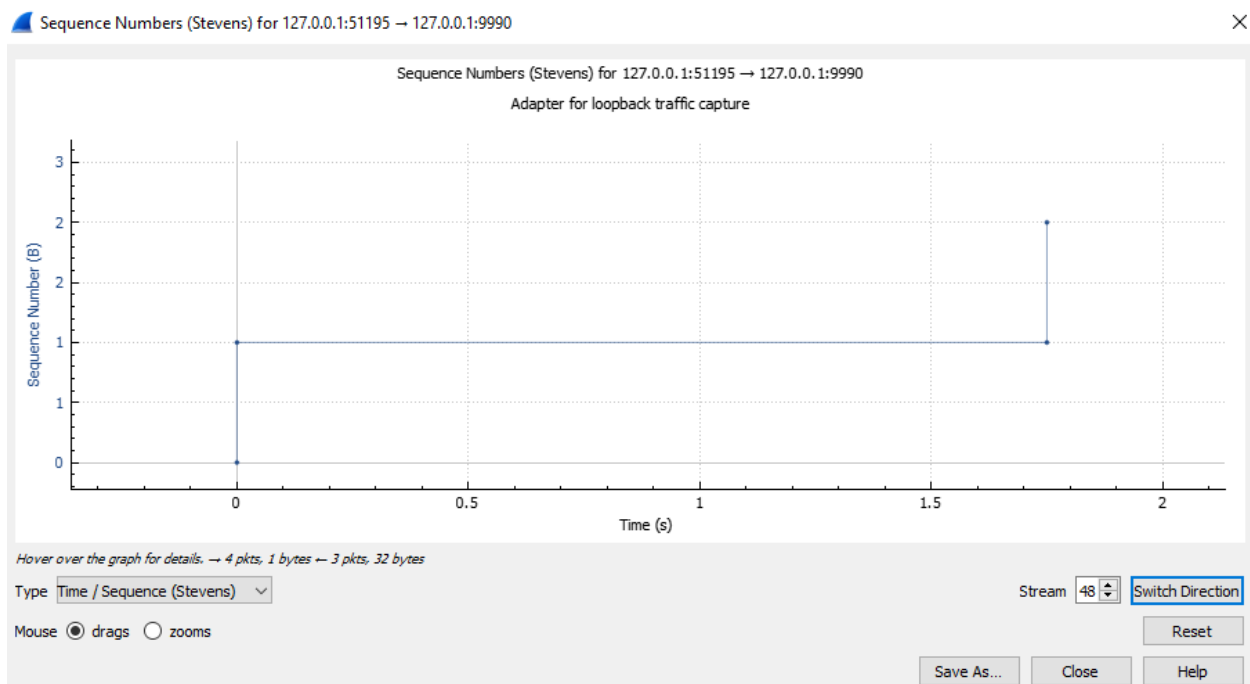
```

Transmission Control Protocol, Src Port: 51196, Dst Port: 49350, Seq: 41090, Ack: 2, Len: 0
  Source Port: 51196
  Destination Port: 49350
  [Stream index: 49]
  [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 0]
  Sequence Number: 41090 (relative sequence number)

```

Throughput represents the number of bytes transferred per time. The number of bytes transferred is the last segment number minus the first segment number, thus $41090 - 1 = 41089$ bytes. The 'per time' can be calculated based on the difference between the first and last segment time which is 1.750295 seconds. The throughput is then $41089 / 1.750295 = 23475.4712777$ bytes per second.

- 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.



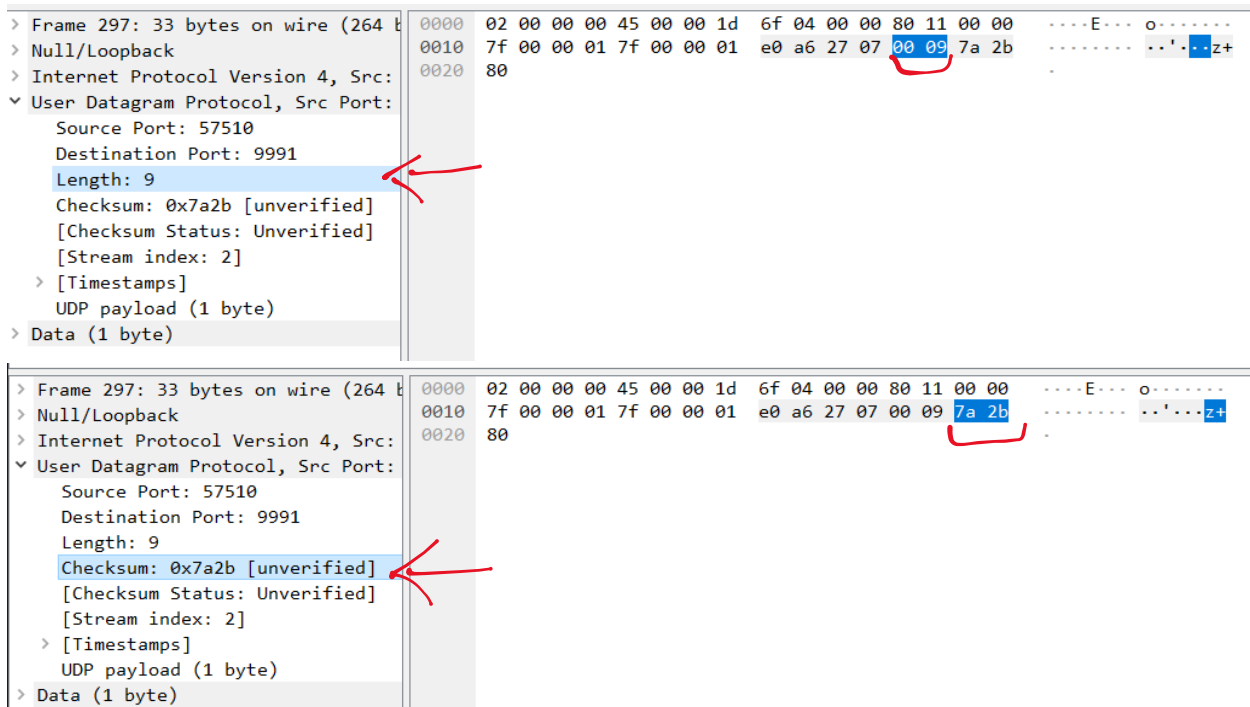
For UDP:

- Select one UDP packet from your trace. From this packet, determine how many fields there are in the UDP header. Name these fields.

udp						
No.	Time	Source	Destination	Protocol	Length	Info
291	17.177018	192.168.56.1	239.255.255.250	SSDP	207	M-SEARCH * HTTP/1.1
292	17.177056	192.168.1.24	239.255.255.250	SSDP	207	M-SEARCH * HTTP/1.1
297	17.505287	127.0.0.1	127.0.0.1	UDP	33	57510 → 9991 Len=1
298	17.505343	127.0.0.1	127.0.0.1	UDP	32	57510 → 9991 Len=0
299	17.505388	127.0.0.1	127.0.0.1	UDP	64	9991 → 57510 Len=32
300	18.181631	192.168.56.1	239.255.255.250	SSDP	207	M-SEARCH * HTTP/1.1
301	18.181658	192.168.1.24	239.255.255.250	SSDP	207	M-SEARCH * HTTP/1.1
302	19.186941	192.168.56.1	239.255.255.250	SSDP	207	M-SEARCH * HTTP/1.1
303	19.186965	192.168.1.24	239.255.255.250	SSDP	207	M-SEARCH * HTTP/1.1
304	19.324058	127.0.0.1	127.0.0.1	UDP	33	57510 → 9991 Len=1
305	19.324085	127.0.0.1	127.0.0.1	UDP	32	57510 → 9991 Len=0
306	19.324118	127.0.0.1	127.0.0.1	UDP	64	9991 → 57510 Len=32
307	20.199749	192.168.56.1	239.255.255.250	SSDP	207	M-SEARCH * HTTP/1.1
308	20.199773	192.168.1.24	239.255.255.250	SSDP	207	M-SEARCH * HTTP/1.1

```
> Frame 297: 33 bytes on wire (264 bits), 33 bytes captured (264 bits) on interface \Device\NPF_{Loopback}, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 57510, Dst Port: 9991
  Source Port: 57510
  Destination Port: 9991
  Length: 9
  Checksum: 0x7a2b [unverified]
  [Checksum Status: Unverified]
  [Stream index: 2]
  > [Timestamps]
  UDP payload (1 byte)
> Data (1 byte)
```

> Frame 297: 33 bytes on wire (264 bits), 33 bytes captured (264 bits) on interface \Device\NPF_{Loopback}, id 0	0000	02 00 00 00 45 00 00 1d	6f 04 00 00 80 11 00 00E...o.....
> Null/Loopback	0010	7f 00 00 01 7f 00 00 01	e0 a6 27 07 00 09 7a 2bz+
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0020	80		
> User Datagram Protocol, Src Port: 57510, Dst Port: 9991				
Source Port: 57510				
Destination Port: 9991				
Length: 9				
Checksum: 0x7a2b [unverified]				
[Checksum Status: Unverified]				
[Stream index: 2]				
> [Timestamps]				
UDP payload (1 byte)				
> Data (1 byte)				
> Frame 297: 33 bytes on wire (264 bits), 33 bytes captured (264 bits) on interface \Device\NPF_{Loopback}, id 0	0000	02 00 00 00 45 00 00 1d	6f 04 00 00 80 11 00 00E...o.....
> Null/Loopback	0010	7f 00 00 01 7f 00 00 01	e0 a6 27 07 00 09 7a 2bz+
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0020	80		
> User Datagram Protocol, Src Port: 57510, Dst Port: 9991				
Source Port: 57510				
Destination Port: 9991				
Length: 9				
Checksum: 0x7a2b [unverified]				
[Checksum Status: Unverified]				
[Stream index: 2]				
> [Timestamps]				
UDP payload (1 byte)				
> Data (1 byte)				

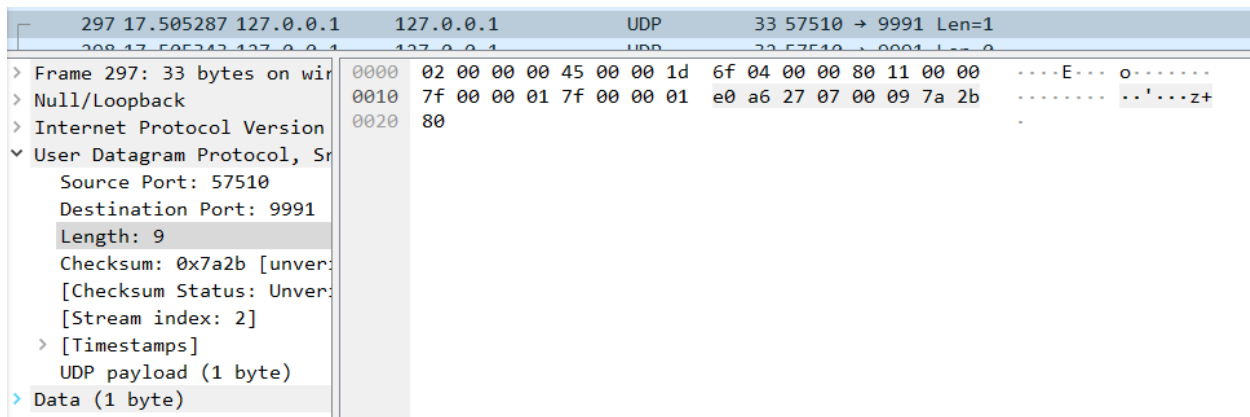


For a UDP packet, the fields in “User Datagram Protocol” are the source port, the destination port, the length and the checksum.

- By consulting the displayed information in Wireshark’s packet content field for this packet, determine the length (in bytes) of each of the UDP header fields

Based on the four images above, the length in bytes of each of the UDP header fields 2 bytes. This is because each field takes up 4 hexadecimal (4 bits), meaning 4 times 4 bits or 16 bits which is 2 bytes. In other words, each field is 2 bytes and the entire UDP header is 8 bytes.

- The value in the Length field is the length of what? Verify your claim with your captured UDP packet.



299	17.505388	127.0.0.1	127.0.0.1	UDP	64	9991 → 57510	Len=32
>	Frame 299:	64 bytes on wire	0000	02 00 00 00 45 00 00 3c	6f 06 00 00 80 11 00 00	----	E--< o.....
>	Null/Loopback		0010	7f 00 00 01 7f 00 00 01	27 07 e0 a6 00 28 88 dd	'...-(!..
>	Internet Protocol Version		0020	df 62 79 65 20 63 68 61	6e 67 65 20 67 65 74 20	-bye cha	nge get
>	User Datagram Protocol, Src Port: 9991		0030	68 65 6c 70 20 70 75 74	20 73 75 6d 6d 61 72 79	help put	summary
>	Destination Port: 57510						
>	Length: 40						
>	Checksum: 0x88dd [unverified]						
>	[Checksum Status: Unverified]						
>	[Stream index: 2]						
>	[Timestamps]						
>	UDP payload (32 bytes)						
>	Data (32 bytes)						

The field “length” represents the total number of bytes in the entire UDP segment (header + payload). This can be confirmed by trying to find the number of bytes in the UDP header and see if it is equal to 8 bytes.

For the first image, length is equal to 9 and the UDP payload is 1 byte. This is expected as this means the UDP header takes up 8 bytes as mentioned in question 2. The calculation is $9-1=8$ bytes. We can also do the same for the second image where the length is 40 and the payload is 32. This means that the UDP header is $40-32=8$ bytes which is also the expected answer.

Implementation Design of the Client Program:

-Code Structure:

- Function Prototype: handle server response
- Function Prototype: send data to server
- Functions Prototype: put, get, change, summary, help, bye commands
- Prompt user information
- Setup client connection (TCP or UDP)
- Loop to allow user to perform commands

Implementation Design of the Server Program:

-Code Structure:

- Function Prototype: handle client request
- Prompt user information
- Setup server connection (TCP or UDP)