

=====

Assignment 3: TCP Sockets Submission:

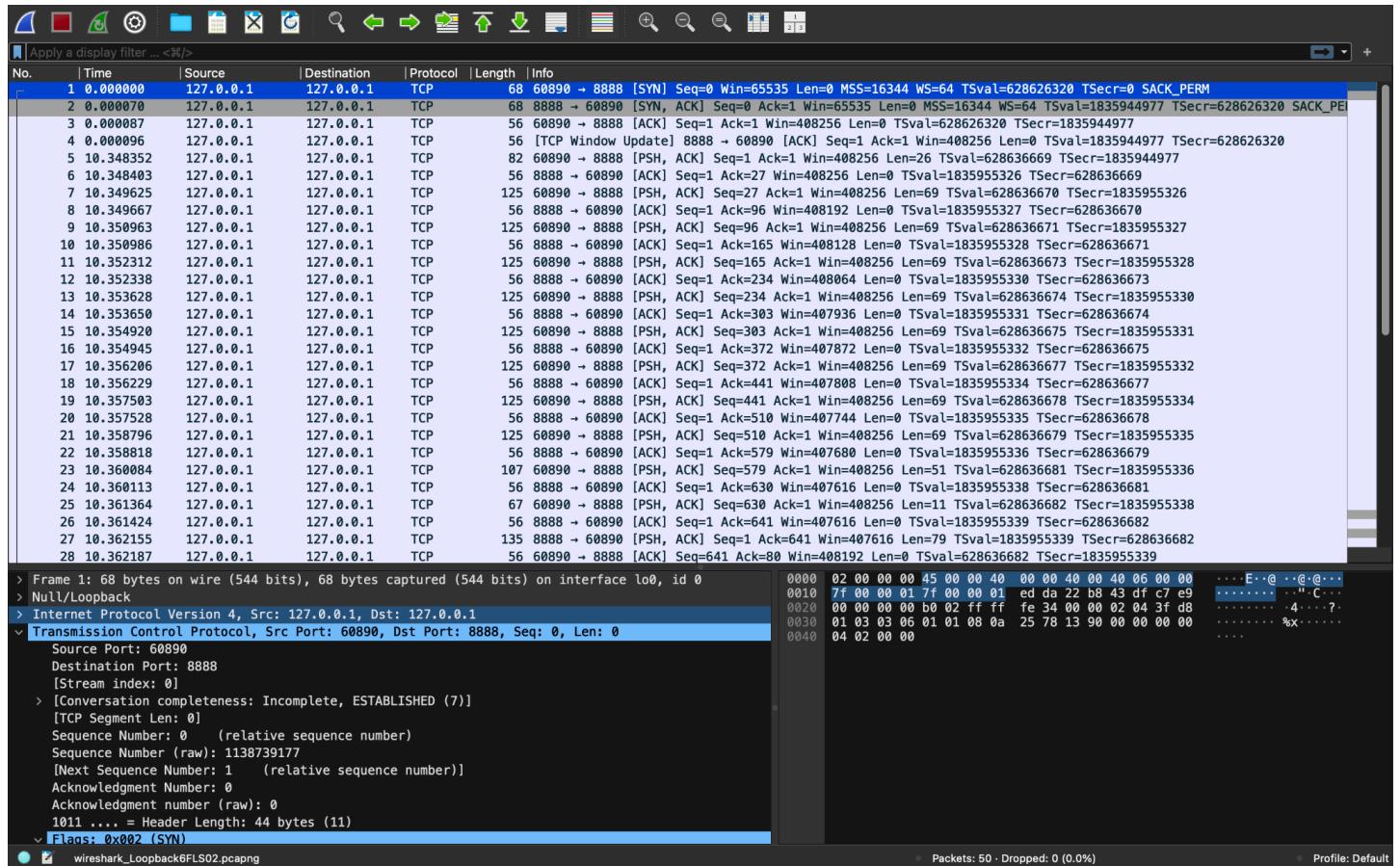
Name: Chandrash Singh

Roll number: 22CS30017

Link of the pcap file: [TCP_packets](#)

=====

1. What are the source and destination IP addresses and ports? Share the screenshots to justify your answer.



From the above screenshot, we can see the first SYN packet where:

- Source (client):
 - Address: **127.0.0.1**
 - Port: **60890**
- Destination (server):
 - Address: **127.0.0.1**
 - Port: **8888**

2. Inspect the Three-way handshaking procedure and capture all packets exchanged for it. Attach the necessary screenshots to demonstrate it

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	68	60890 → 8888 [SYN] Seq=0 Win=65535 Len=0 MSS=16344 WS=64 TSval=628626320 TSecr=0 SACK_PERM
2	0.000070	127.0.0.1	127.0.0.1	TCP	68	8888 → 60890 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=16344 WS=64 TSval=1835944977 TSecr=628626320 SACK_PERM
3	0.000087	127.0.0.1	127.0.0.1	TCP	56	60890 → 8888 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=628626320 TSecr=1835944977
4	0.000096	127.0.0.1	127.0.0.1	TCP	56	[TCP Window Update] 8888 → 60890 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=1835944977 TSecr=628626320
5	0.348352	127.0.0.1	127.0.0.1	TCP	82	60890 → 8888 [PSH, ACK] Seq=1 Ack=1 Win=408256 Len=26 TSval=628636669 TSecr=1835944977
6	0.348403	127.0.0.1	127.0.0.1	TCP	56	8888 → 60890 [ACK] Seq=1 Ack=27 Win=408256 Len=0 TSval=1835955326 TSecr=628636669
7	0.349625	127.0.0.1	127.0.0.1	TCP	125	60890 → 8888 [PSH, ACK] Seq=27 Ack=1 Win=408256 Len=69 TSval=628636670 TSecr=1835955326
8	0.349667	127.0.0.1	127.0.0.1	TCP	56	8888 → 60890 [ACK] Seq=1 Ack=96 Win=408192 Len=0 TSval=1835955327 TSecr=628636670
9	0.350963	127.0.0.1	127.0.0.1	TCP	125	60890 → 8888 [PSH, ACK] Seq=96 Ack=1 Win=408256 Len=69 TSval=628636671 TSecr=1835955327
10	0.350986	127.0.0.1	127.0.0.1	TCP	56	8888 → 60890 [ACK] Seq=1 Ack=165 Win=408128 Len=0 TSval=1835955328 TSecr=628636671
11	0.352312	127.0.0.1	127.0.0.1	TCP	125	60890 → 8888 [PSH, ACK] Seq=165 Ack=1 Win=408256 Len=69 TSval=628636673 TSecr=1835955328
12	0.352338	127.0.0.1	127.0.0.1	TCP	56	8888 → 60890 [ACK] Seq=1 Ack=234 Win=408064 Len=0 TSval=1835955330 TSecr=628636673
13	0.353628	127.0.0.1	127.0.0.1	TCP	125	60890 → 8888 [PSH, ACK] Seq=234 Ack=1 Win=408256 Len=69 TSval=628636674 TSecr=1835955330
14	0.353650	127.0.0.1	127.0.0.1	TCP	56	8888 → 60890 [ACK] Seq=1 Ack=303 Win=407936 Len=0 TSval=1835955331 TSecr=628636674
15	0.354920	127.0.0.1	127.0.0.1	TCP	125	60890 → 8888 [PSH, ACK] Seq=303 Ack=1 Win=408256 Len=69 TSval=628636675 TSecr=1835955331
16	0.354945	127.0.0.1	127.0.0.1	TCP	56	8888 → 60890 [ACK] Seq=1 Ack=372 Win=407872 Len=0 TSval=1835955332 TSecr=628636675
17	0.356206	127.0.0.1	127.0.0.1	TCP	125	60890 → 8888 [PSH, ACK] Seq=372 Ack=1 Win=408256 Len=69 TSval=628636677 TSecr=1835955332
18	0.356229	127.0.0.1	127.0.0.1	TCP	56	8888 → 60890 [ACK] Seq=1 Ack=441 Win=407808 Len=0 TSval=1835955334 TSecr=628636677
19	0.357503	127.0.0.1	127.0.0.1	TCP	125	60890 → 8888 [PSH, ACK] Seq=441 Ack=1 Win=408256 Len=61 TSval=628636678 TSecr=1835955334
20	0.357528	127.0.0.1	127.0.0.1	TCP	56	8888 → 60890 [ACK] Seq=1 Ack=510 Win=407744 Len=0 TSval=1835955335 TSecr=628636678
21	0.358796	127.0.0.1	127.0.0.1	TCP	125	60890 → 8888 [PSH, ACK] Seq=510 Ack=1 Win=408256 Len=69 TSval=628636679 TSecr=1835955335
22	0.358818	127.0.0.1	127.0.0.1	TCP	56	8888 → 60890 [ACK] Seq=1 Ack=579 Win=407680 Len=0 TSval=1835955336 TSecr=628636679
23	0.360084	127.0.0.1	127.0.0.1	TCP	107	60890 → 8888 [PSH, ACK] Seq=579 Ack=1 Win=408256 Len=51 TSval=628636681 TSecr=1835955336
24	0.360113	127.0.0.1	127.0.0.1	TCP	56	8888 → 60890 [ACK] Seq=1 Ack=630 Win=407616 Len=0 TSval=1835955338 TSecr=628636681
25	0.361364	127.0.0.1	127.0.0.1	TCP	67	60890 → 8888 [PSH, ACK] Seq=630 Ack=1 Win=408256 Len=11 TSval=628636682 TSecr=1835955338
26	0.361424	127.0.0.1	127.0.0.1	TCP	56	8888 → 60890 [ACK] Seq=1 Ack=641 Win=407616 Len=0 TSval=1835955339 TSecr=628636682
27	0.362155	127.0.0.1	127.0.0.1	TCP	135	8888 → 60890 [PSH, ACK] Seq=1 Ack=641 Win=407616 Len=79 TSval=1835955339 TSecr=628636682
28	0.362187	127.0.0.1	127.0.0.1	TCP	56	60890 → 8888 [ACK] Seq=641 Ack=89 Win=408192 Len=0 TSval=628636682 TSecr=1835955339

Three-way Handshake:

- Look for the **first three** packets in the above screenshot:
 - SYN (Client → Server)
 - SYN-ACK (Server → Client)
 - ACK (Client → Server)

3. Inspect the connection closure procedure and capture all packets exchanged for it. Attach the necessary screenshots to demonstrate it.

Frame 47: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface lo0, id 0

Null/Loopback

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

Transmission Control Protocol, Src Port: 60890, Dst Port: 8888, Seq: 643, Ack: 615, Len: 0

Source Port: 60890
Destination Port: 8888
[Stream index: 0]
[Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 0]
Sequence Number: 643 (relative sequence number)
Sequence Number (raw): 1138739820
[Next Sequence Number: 644 (relative sequence number)]
Acknowledgment Number: 615 (relative ack number)
Acknowledgment header number (raw): 2003744975
1000 = Header Length: 32 bytes (8)
Flags: 0x011 (FIN, ACK)

0000 02 00 00 00 45 00 00 34 00 00 40 00 00 00 00 00 ... E..4 ..@..@..
0010 7f 00 00 01 7f 00 00 01 ed 2a b8 43 df ca 6c .."C..l
0020 77 6e b8 cf 80 11 18 e2 fe 28 00 00 01 01 08 0a wn(.....
0030 25 78 44 e7 6d 6e 7d 68 %xD..mn)h

We can find the sequence **at the end**:

- FIN (Client → Server) → **No.47**
 - ACK (Server → Client) → **No.48**
 - FIN (Server → Client) → **No.49**
 - ACK (Client → Server) → **No.50**

4. Inspect the traffic and count the number of packets exchanged for the transfer of a file(related to data only) between client and server. Plot a graph ‘file size vs the number of packets’ clearly based on your observation.

To count data packets, I used:

- Filter: `tcp.port == 8888 && tcp.len > 0`
- This shows only packets containing data

Note:

- Client sends file in chunks of max size = 70 bytes
- Server sends file in chunks of max size = 80 bytes

But for receiving a chunk, both use a buffer of size 100 bytes, because we were stated that chuck size will not be greater than 100 bytes. So, whatever be the size of the chunk used to send, the other server/client can receive it in a single packet.

No.	Time	Source	Destination	Protocol	Length	Info
5	10.348352	127.0.0.1	127.0.0.1	TCP	82	60890 → 8888 [PSH, ACK] Seq=1 Ack=1 Win=408256 Len=26 TSval=628636669 TSecr=1835944977
7	10.349625	127.0.0.1	127.0.0.1	TCP	125	60890 → 8888 [PSH, ACK] Seq=27 Ack=1 Win=408256 Len=69 TSval=628636670 TSecr=1835955326
9	10.350963	127.0.0.1	127.0.0.1	TCP	125	60890 → 8888 [PSH, ACK] Seq=96 Ack=1 Win=408256 Len=69 TSval=628636671 TSecr=1835955327
11	10.352312	127.0.0.1	127.0.0.1	TCP	125	60890 → 8888 [PSH, ACK] Seq=165 Ack=1 Win=408256 Len=69 TSval=628636673 TSecr=1835955328
13	10.353628	127.0.0.1	127.0.0.1	TCP	125	60890 → 8888 [PSH, ACK] Seq=234 Ack=1 Win=408256 Len=69 TSval=628636674 TSecr=1835955330
15	10.354920	127.0.0.1	127.0.0.1	TCP	125	60890 → 8888 [PSH, ACK] Seq=303 Ack=1 Win=408256 Len=69 TSval=628636675 TSecr=1835955331
17	10.356206	127.0.0.1	127.0.0.1	TCP	125	60890 → 8888 [PSH, ACK] Seq=372 Ack=1 Win=408256 Len=69 TSval=628636677 TSecr=1835955332
19	10.357503	127.0.0.1	127.0.0.1	TCP	125	60890 → 8888 [PSH, ACK] Seq=441 Ack=1 Win=408256 Len=69 TSval=628636678 TSecr=1835955334
21	10.358796	127.0.0.1	127.0.0.1	TCP	125	60890 → 8888 [PSH, ACK] Seq=510 Ack=1 Win=408256 Len=69 TSval=628636679 TSecr=1835955335
23	10.360084	127.0.0.1	127.0.0.1	TCP	107	60890 → 8888 [PSH, ACK] Seq=579 Ack=1 Win=408256 Len=51 TSval=628636681 TSecr=1835955336
25	10.361364	127.0.0.1	127.0.0.1	TCP	67	60890 → 8888 [PSH, ACK] Seq=630 Ack=1 Win=408256 Len=11 TSval=628636682 TSecr=1835955338
27	10.362155	127.0.0.1	127.0.0.1	TCP	135	8888 → 60890 [PSH, ACK] Seq=1 Ack=641 Win=407616 Len=79 TSval=1835955339 TSecr=628636682
29	10.363310	127.0.0.1	127.0.0.1	TCP	135	8888 → 60890 [PSH, ACK] Seq=80 Ack=641 Win=407616 Len=79 TSval=1835955341 TSecr=628636682
31	10.364463	127.0.0.1	127.0.0.1	TCP	135	8888 → 60890 [PSH, ACK] Seq=159 Ack=641 Win=407616 Len=79 TSval=1835955342 TSecr=628636682
33	10.365616	127.0.0.1	127.0.0.1	TCP	135	8888 → 60890 [PSH, ACK] Seq=238 Ack=641 Win=407616 Len=79 TSval=1835955343 TSecr=628636685
35	10.366766	127.0.0.1	127.0.0.1	TCP	135	8888 → 60890 [PSH, ACK] Seq=317 Ack=641 Win=407616 Len=79 TSval=1835955344 TSecr=628636686
37	10.367918	127.0.0.1	127.0.0.1	TCP	135	8888 → 60890 [PSH, ACK] Seq=396 Ack=641 Win=407616 Len=79 TSval=1835955345 TSecr=628636687
39	10.369069	127.0.0.1	127.0.0.1	TCP	135	8888 → 60890 [PSH, ACK] Seq=475 Ack=641 Win=407616 Len=79 TSval=1835955347 TSecr=628636688
41	10.370232	127.0.0.1	127.0.0.1	TCP	106	8888 → 60890 [PSH, ACK] Seq=554 Ack=641 Win=407616 Len=51 TSval=1835955348 TSecr=628636690
43	10.371391	127.0.0.1	127.0.0.1	TCP	67	8888 → 60890 [PSH, ACK] Seq=604 Ack=641 Win=407616 Len=11 TSval=1835955349 TSecr=628636691
45	12.630198	127.0.0.1	127.0.0.1	TCP	58	60890 → 8888 [PSH, ACK] Seq=641 Ack=615 Win=407680 Len=2 TSval=628638951 TSecr=1835955349

> Frame 5: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface lo0, id 0	0000 02 00 00 00 45 00 00 4e 00 00 40 00 40 06 00 00	...E-N:@@...
> Null/Loopback	0010 7f 00 00 01 7f 00 00 01 ed da 22 b8 43 df c7 ea	wn i...B...
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0020 77 6e b6 69 80 18 18 eb fe 42 00 00 01 01 08 0a	%x:-mnL DEFPRTVW
Transmission Control Protocol, Src Port: 60890, Dst Port: 8888, Seq: 1, Ack: 1, Len: 26	0030 25 78 3b fd 6d 6e 4c 11 44 45 46 50 52 54 56 57	LMZAYGHQ SIUJXKB
Source Port: 60890	0040 4c 4d 5a 41 59 47 48 51 53 49 55 4a 58 4b 42 43	NO
Destination Port: 8888	0050 4e 4f	
[Stream index: 0]		
> [Conversation completeness: Complete, WITH_DATA (31)]		
[TCP Segment Len: 26]		
Sequence Number: 1 (relative sequence number)		
Sequence Number (raw): 1138739178		
[Next Sequence Number: 27 (relative sequence number)]		
Acknowledgment Number: 1 (relative ack number)		
Acknowledgment number (raw): 2003744361		
1000 = Header Length: 32 bytes (8)		
Flags: 0x018 (PSH, ACK)		

In the above screenshot,

- No.5 is the KEY packet
- From No.7 to No.23 are the packets from client to server of the original file
- No.25 is the END_OF_FILE
- From No.27 to No.41 are the packets from server to client of the encrypted file
- No.43 is the END_OF_FILE
- No.45 is the “No” from the client to close the connection

Total number of packets exchanged for the transfer of a file(related to data only) between client and server:

$$9 \text{ (from client to server)} + 8 \text{ (from server to client)} = 17 \text{ packets}$$

File size = sum of the data size of each packet sent by client = sum of data size of each packet sent by server

$$= 69 + 69 + 69 + 69 + 69 + 69 + 69 + 51 = 603 \text{ bytes}$$

$$= 79 + 79 + 79 + 79 + 79 + 79 + 50 = 603 \text{ bytes}$$

I have also implemented the function to print the chunk number and the size of each chunk:
So the overall statistics can also be seen from the terminal

Terminal

```
~/.Doc/6/CN/LAB/LA3 main i3 ?1 > ls 04:22:41 pm
CS39006_Networks_Lab_Assignment3.pdf readme.md
TCP_packets.pcapng          retrieveencfileclient.c
doencfilesrvr.c              sample.txt
key.txt                      sample2.txt
makefile                     tut
~/.Doc/6/CN/LAB/LA3 main i3 ?1 > cat sample2.txt 04:22:44 pm
A socketfd file descriptor is an integer that uniquely identifies a socket within a process. This descriptor is used to perform operations on the socket such as reading from or writing to it. When creating a server the socket function is called to initialize a socketfd. This descriptor is then used with other functions like bind, listen and accept to set up the server and manage incoming connections

The socketfd is crucial for managing communication over the network and is treated similarly to other file descriptors in Unix-like systems, allowing operations like read and write to be used on sockets as well.
~/.Doc/6/CN/LAB/LA3 main i3 ?1 > make runsv 04:22:49 pm
gcc -Wall -o svr doencfilesrvr.c
./svr
Server listening on port 8888...
+++ New client connected: 127.0.0.1:60890
[127.0.0.1:60890] chunk #1: 69 bytes
[127.0.0.1:60890] chunk #2: 69 bytes
[127.0.0.1:60890] chunk #3: 69 bytes
[127.0.0.1:60890] chunk #4: 69 bytes
[127.0.0.1:60890] chunk #5: 69 bytes
[127.0.0.1:60890] chunk #6: 69 bytes
[127.0.0.1:60890] chunk #7: 69 bytes
[127.0.0.1:60890] chunk #8: 69 bytes
[127.0.0.1:60890] chunk #9: 51 bytes
--> Total chunks received: 9, total bytes: 603

[127.0.0.1:60890] chunk #1: 79 bytes
[127.0.0.1:60890] chunk #2: 79 bytes
[127.0.0.1:60890] chunk #3: 79 bytes
[127.0.0.1:60890] chunk #4: 79 bytes
[127.0.0.1:60890] chunk #5: 79 bytes
[127.0.0.1:60890] chunk #6: 79 bytes
[127.0.0.1:60890] chunk #7: 79 bytes
[127.0.0.1:60890] chunk #8: 50 bytes
--> Total chunks sent: 8, total bytes: 603

-----
Client disconnected from 127.0.0.1:60890
[]

~/.Doc/6/CN/LAB/LA3 main i3 ?1 > make runcl 04:22:29 pm
gcc -Wall -o cli retrieveencfileclient.c
./cli

Enter filename to encrypt: sample2.txt
Enter 26-character encryption key: DEFPRTVWLMZAYGHQSIUJXKBCNO
[Client] sent chunk #1: 69 bytes
[Client] sent chunk #2: 69 bytes
[Client] sent chunk #3: 69 bytes
[Client] sent chunk #4: 69 bytes
[Client] sent chunk #5: 69 bytes
[Client] sent chunk #6: 69 bytes
[Client] sent chunk #7: 69 bytes
[Client] sent chunk #8: 69 bytes
[Client] sent chunk #9: 51 bytes
--> Total chunks sent: 9, total bytes: 603

[Client] received chunk #1: 79 bytes
[Client] received chunk #2: 79 bytes
[Client] received chunk #3: 79 bytes
[Client] received chunk #4: 79 bytes
[Client] received chunk #5: 79 bytes
[Client] received chunk #6: 79 bytes
[Client] received chunk #7: 79 bytes
[Client] received chunk #8: 50 bytes
--> Total chunks received: 8, total bytes: 603

File encrypted successfully!
Original file: sample2.txt
Encrypted file: sample2.txt.enc
-----

Encrypt another file? (Yes/No): No
~/.Doc/6/CN/LAB/LA3 main i3 ?6 > cat sample2.txt.enc 13s 04:23:28 pm
D uhfztp tlar pruflqjhi lu dg lgjrvri jwdj xglxran lprgjltlru d uhfzrj bljwlg d qihfru
u Jwlw pruflqjhi lu xurp jh grithiy hqridjlhg u hqridjlhg u hqridjlhg u hqridjlhg u hqridjlhg
ljlgv jh lj Bwrg firdjlgd d urikri jwr uhfzrj txgfjlhg lu fdaarp jh lgjldalor d uhfztp
Jwlw pruflqjhi lu jwrg xurp bljw hjwri txgfjlhg alrz elgp alujrg dpp dffrjq jh urj xq
jwr urikri dgp ydgdrv lgfhylgv fhngrffjlhg

Jwr uhfztp lu fixflda thi ydgdrv lgfhylgv fhyyxglfdjlg hkrj grjbhiz dgp lu jirdjpr ulylad
an jh hjwri tlar pruflqjhi lg xglc alrz unujry daahblgv hqridjlhg alrz irdp dgp bilj
r jh er xurp hg uhfzrju du braa
~/.Doc/6/CN/LAB/LA3 main i3 ?6 > [ ] 04:23:52 pm
```

Plot a graph 'file size vs the number of packets' clearly based on your observation:

```
[127.0.0.1:60890] chunk #1: 69 bytes
[127.0.0.1:60890] chunk #2: 69 bytes
[127.0.0.1:60890] chunk #3: 69 bytes
[127.0.0.1:60890] chunk #4: 69 bytes
[127.0.0.1:60890] chunk #5: 69 bytes
[127.0.0.1:60890] chunk #6: 69 bytes
[127.0.0.1:60890] chunk #7: 69 bytes
[127.0.0.1:60890] chunk #8: 69 bytes
[127.0.0.1:60890] chunk #9: 51 bytes
--> Total chunks received: 9, total bytes: 603

[127.0.0.1:60890] chunk #1: 79 bytes
[127.0.0.1:60890] chunk #2: 79 bytes
[127.0.0.1:60890] chunk #3: 79 bytes
[127.0.0.1:60890] chunk #4: 79 bytes
[127.0.0.1:60890] chunk #5: 79 bytes
[127.0.0.1:60890] chunk #6: 79 bytes
[127.0.0.1:60890] chunk #7: 79 bytes
[127.0.0.1:60890] chunk #8: 50 bytes
--> Total chunks sent: 8, total bytes: 603

-----
Client disconnected from 127.0.0.1:60890

+++ New client connected: 127.0.0.1:60964
[127.0.0.1:60964] chunk #1: 69 bytes
[127.0.0.1:60964] chunk #2: 69 bytes
[127.0.0.1:60964] chunk #3: 69 bytes
[127.0.0.1:60964] chunk #4: 69 bytes
[127.0.0.1:60964] chunk #5: 69 bytes
[127.0.0.1:60964] chunk #6: 69 bytes
[127.0.0.1:60964] chunk #7: 53 bytes
--> Total chunks received: 7, total bytes: 467

[127.0.0.1:60964] chunk #1: 79 bytes
[127.0.0.1:60964] chunk #2: 79 bytes
[127.0.0.1:60964] chunk #3: 79 bytes
[127.0.0.1:60964] chunk #4: 79 bytes
[127.0.0.1:60964] chunk #5: 79 bytes
[127.0.0.1:60964] chunk #6: 72 bytes
--> Total chunks sent: 6, total bytes: 467

-----
[127.0.0.1:60964] chunk #1: 27 bytes
--> Total chunks received: 1, total bytes: 27

[127.0.0.1:60964] chunk #1: 27 bytes
--> Total chunks sent: 1, total bytes: 27

-----
Client disconnected from 127.0.0.1:60964
[]

-----
```

..em/CN/LAB/LA3 (-zsh)
jwr userkri dgp ydgdrv lgfhylgv fhggrfjlhgu
Jwr uhfztp lu fixflda thi ydgdrvlgv fhyyxglfdjhlg hkri jwr grjbhiz dgp lu jirdjpr ulyladi
an jh hjwri tlar pruiflqjhui lg Xgcl alrz unujry daahblgv hqrjdjlhgu alrz irdp dgp bilr
r jh er xurp hg uhfztp du braa
~/Doc/6/CN/LAB/LA3 main i3 76 > █ 04:23:52 pm

..em/CN/LAB/LA3 (-zsh)

Enter filename to encrypt: sample3.txt
Enter 26-character encryption key: DEFPRPTVWLMZAYGHQSIUJXKBCNO
[Client] sent chunk #1: 69 bytes
[Client] sent chunk #2: 69 bytes
[Client] sent chunk #3: 69 bytes
[Client] sent chunk #4: 69 bytes
[Client] sent chunk #5: 69 bytes
[Client] sent chunk #6: 69 bytes
[Client] sent chunk #7: 53 bytes
--> Total chunks sent: 7, total bytes: 467

[Client] received chunk #1: 79 bytes
[Client] received chunk #2: 79 bytes
[Client] received chunk #3: 79 bytes
[Client] received chunk #4: 79 bytes
[Client] received chunk #5: 79 bytes
[Client] received chunk #6: 72 bytes
--> Total chunks received: 6, total bytes: 467

File encrypted successfully!
Original file: sample3.txt
Encrypted file: sample3.txt.enc

Encrypt another file? (Yes/No): Yes

Enter filename to encrypt: sample.txt
Enter 26-character encryption key: DEFPRPTVWLMZAYGHQSIUJXKBCNO
[Client] sent chunk #1: 27 bytes
--> Total chunks sent: 1, total bytes: 27

[Client] received chunk #1: 27 bytes
--> Total chunks received: 1, total bytes: 27

File encrypted successfully!
Original file: sample.txt
Encrypted file: sample.txt.enc

Encrypt another file? (Yes/No): No
~/Doc/6/CN/LAB/LA3 main i3 711 > █ 04:41:12 pm

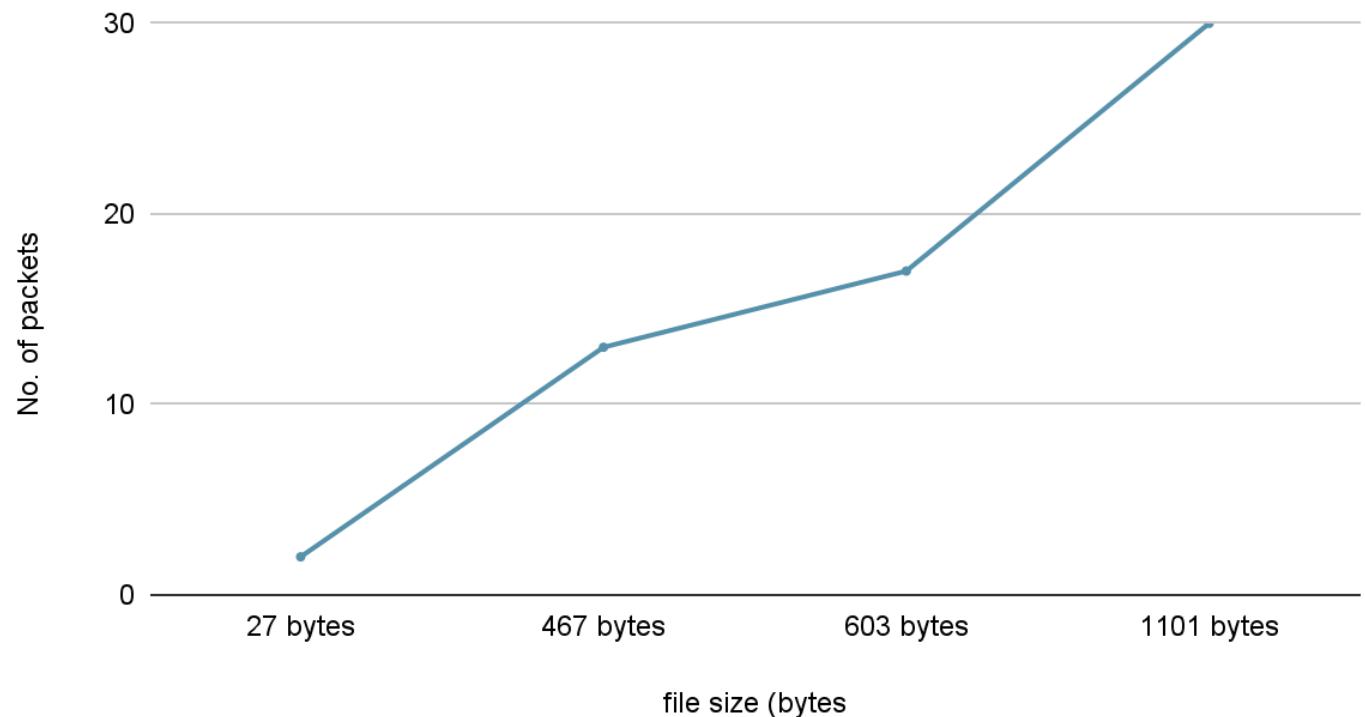
From the above three sample files:

- 1) 603 bytes -> 17 packets
- 2) 467 bytes -> 13 packets
- 3) 27 bytes -> 2 packets
- 4) 1101 bytes -> 30 packets

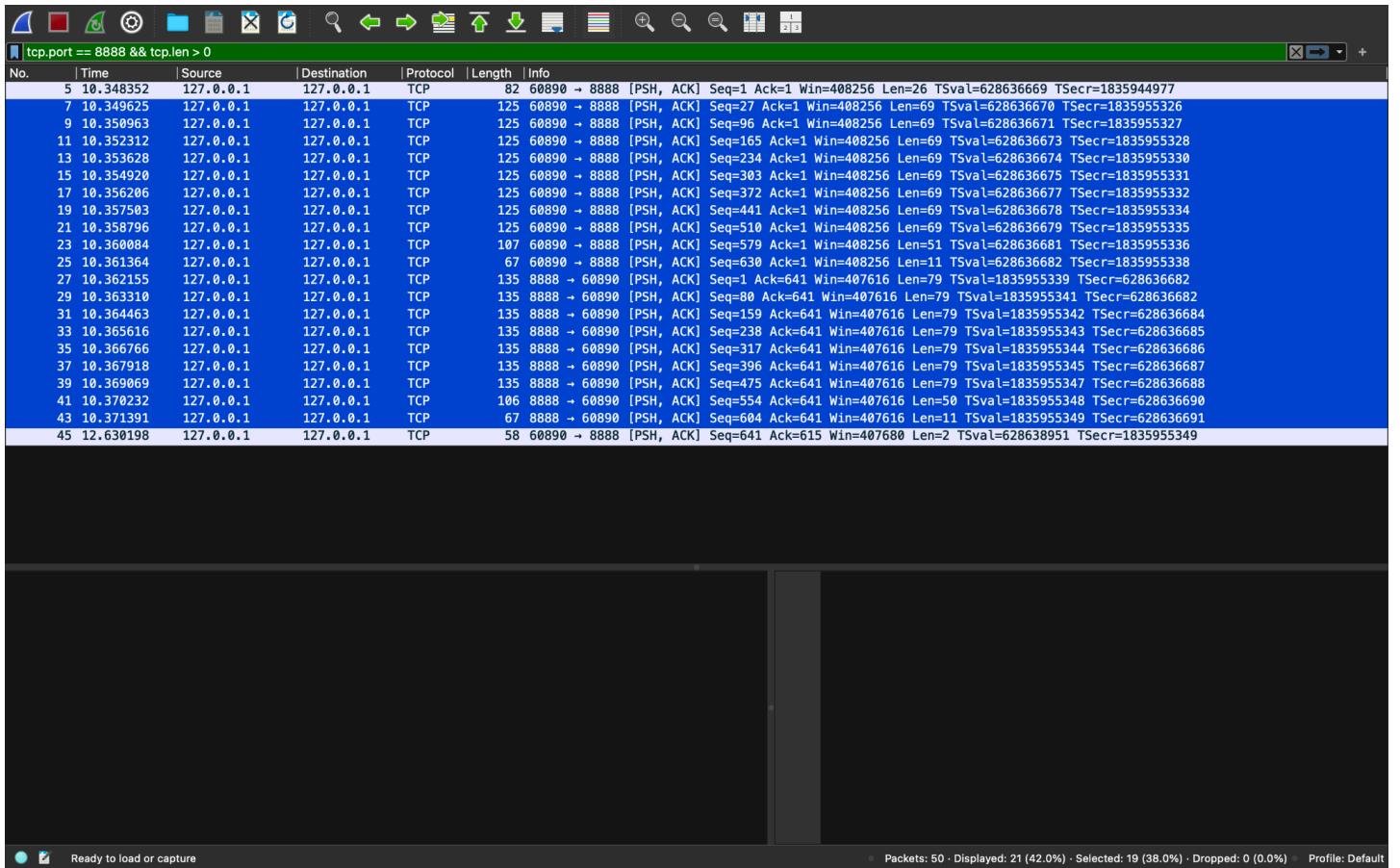
Mathematically: for a file of size B bytes

- Client will send $\lceil B/69 \rceil$ packets
- Server will send $\lceil B/79 \rceil$ packets

No. of packets vs file size

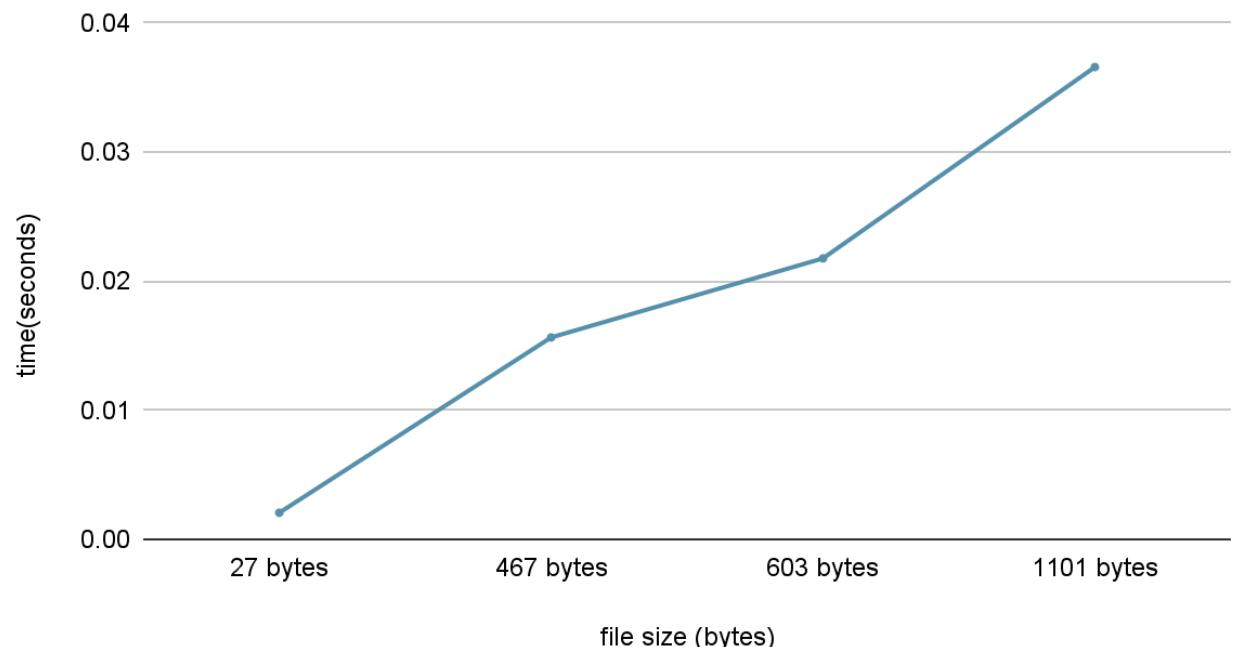


5. Measure the total time taken for the file transfer , its encryption and send it back from server to the client. Plot a graph ‘file size vs time’ clearly based on your observation and also attach the necessary screenshots.



- Total Time taken for the file transfer of size 603 bytes = 10.371391 - 10.349626 = 0.0217652 seconds
- More observations (can be verified with the attached pcap files)
 - 467 bytes → 10.577517 - 10.561878 = 0.015639 seconds
 - 27 bytes → 0.003645 - 0.001589 = 0.002056 seconds
 - 1101 bytes → 0.037686 - 0.001100 = 0.036586 seconds

time(seconds) vs file size



6. Calculate the average size packet exchanged during the data communication? Take reference of the plotted graph in the above question.

Using the Statistics button in wireshark to find the **average packet length**

To count data packets, I used:

- Filter: `tcp.port == 8888 && tcp.len > 0 && tcp.len != 11`
- This shows only packets containing data
- Excluded the END_OF_FILE packet
- Considering the KEY packet in data communication
- So, the count of packets in the below statistics is: no. of packets answered in Q4 + 1(for key)
- As the Q4 asked about the comparison about file size, but here we have data communication

For file size 27 bytes: **82.67 bytes**

Packet Lengths:									
Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start	
Packet Lengths	3	82.67	82	83	0.8230	100%	0.0300	0.000	
0-19	0	-	-	-	0.0000	0.00%	-	-	
20-39	0	-	-	-	0.0000	0.00%	-	-	
40-79	0	-	-	-	0.0000	0.00%	-	-	
80-159	3	82.67	82	83	0.8230	100.00%	0.0300	0.000	
160-319	0	-	-	-	0.0000	0.00%	-	-	
320-639	0	-	-	-	0.0000	0.00%	-	-	
640-1279	0	-	-	-	0.0000	0.00%	-	-	
1280-2559	0	-	-	-	0.0000	0.00%	-	-	
2560-5119	0	-	-	-	0.0000	0.00%	-	-	
5120 and greater	0	-	-	-	0.0000	0.00%	-	-	

For file size 467 bytes: **124.57 bytes**

Packet Lengths:									
Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst	Start
Packet Lengths	14	124.57	82	135	0.8378	100%	0.1400	10.561	
0-19	0	-	-	-	0.0000	0.00%	-	-	
20-39	0	-	-	-	0.0000	0.00%	-	-	
40-79	0	-	-	-	0.0000	0.00%	-	-	
80-159	14	124.57	82	135	0.8378	100.00%	0.1400	10.561	
160-319	0	-	-	-	0.0000	0.00%	-	-	
320-639	0	-	-	-	0.0000	0.00%	-	-	
640-1279	0	-	-	-	0.0000	0.00%	-	-	
1280-2559	0	-	-	-	0.0000	0.00%	-	-	
2560-5119	0	-	-	-	0.0000	0.00%	-	-	
5120 and greater	0	-	-	-	0.0000	0.00%	-	-	

For file size 603 bytes: **124.44 bytes**

Packet Lengths:

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Packet Lengths	18	124.44	82	135	0.8227	100%	0.1800	10.348
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	0	-	-	-	0.0000	0.00%	-	-
80-159	18	124.44	82	135	0.8227	100.00%	0.1800	10.348
160-319	0	-	-	-	0.0000	0.00%	-	-
320-639	0	-	-	-	0.0000	0.00%	-	-
640-1279	0	-	-	-	0.0000	0.00%	-	-
1280-2559	0	-	-	-	0.0000	0.00%	-	-
2560-5119	0	-	-	-	0.0000	0.00%	-	-
5120 and greater	0	-	-	-	0.0000	0.00%	-	-

For file size 1101 bytes: **127.87 bytes**

Packet Lengths:

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Packet Lengths	31	127.87	82	135	0.8226	100%	0.3100	0.000
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	0	-	-	-	0.0000	0.00%	-	-
80-159	31	127.87	82	135	0.8226	100.00%	0.3100	0.000
160-319	0	-	-	-	0.0000	0.00%	-	-
320-639	0	-	-	-	0.0000	0.00%	-	-
640-1279	0	-	-	-	0.0000	0.00%	-	-
1280-2559	0	-	-	-	0.0000	0.00%	-	-
2560-5119	0	-	-	-	0.0000	0.00%	-	-
5120 and greater	0	-	-	-	0.0000	0.00%	-	-

Average packet size vs file size

