

=====

## Assignment 3: TCP Sockets Submission:

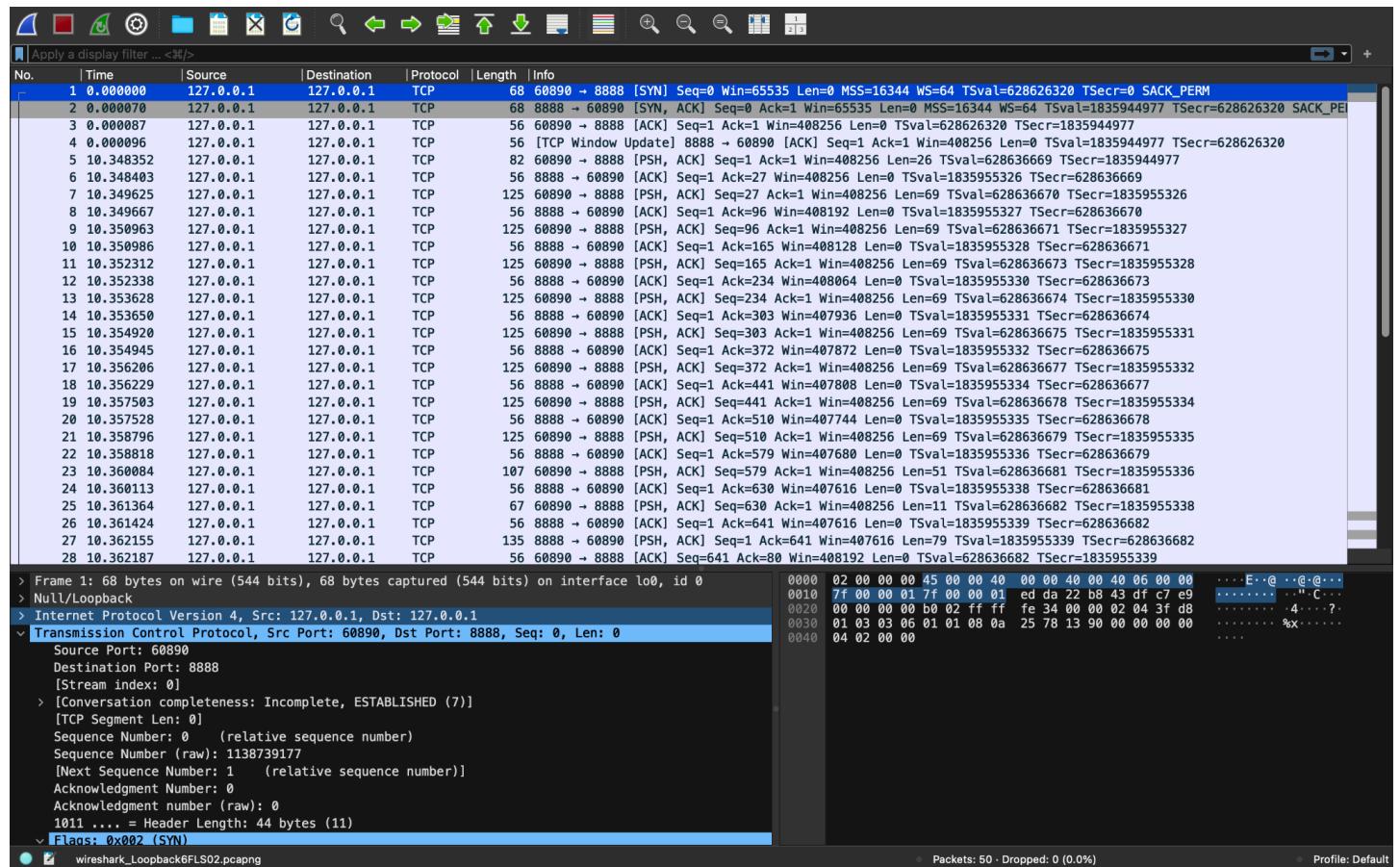
Name: Chandrash Singh

Roll number: 22CS30017

Link of the pcap file: [TCP\\_packets](#)

=====

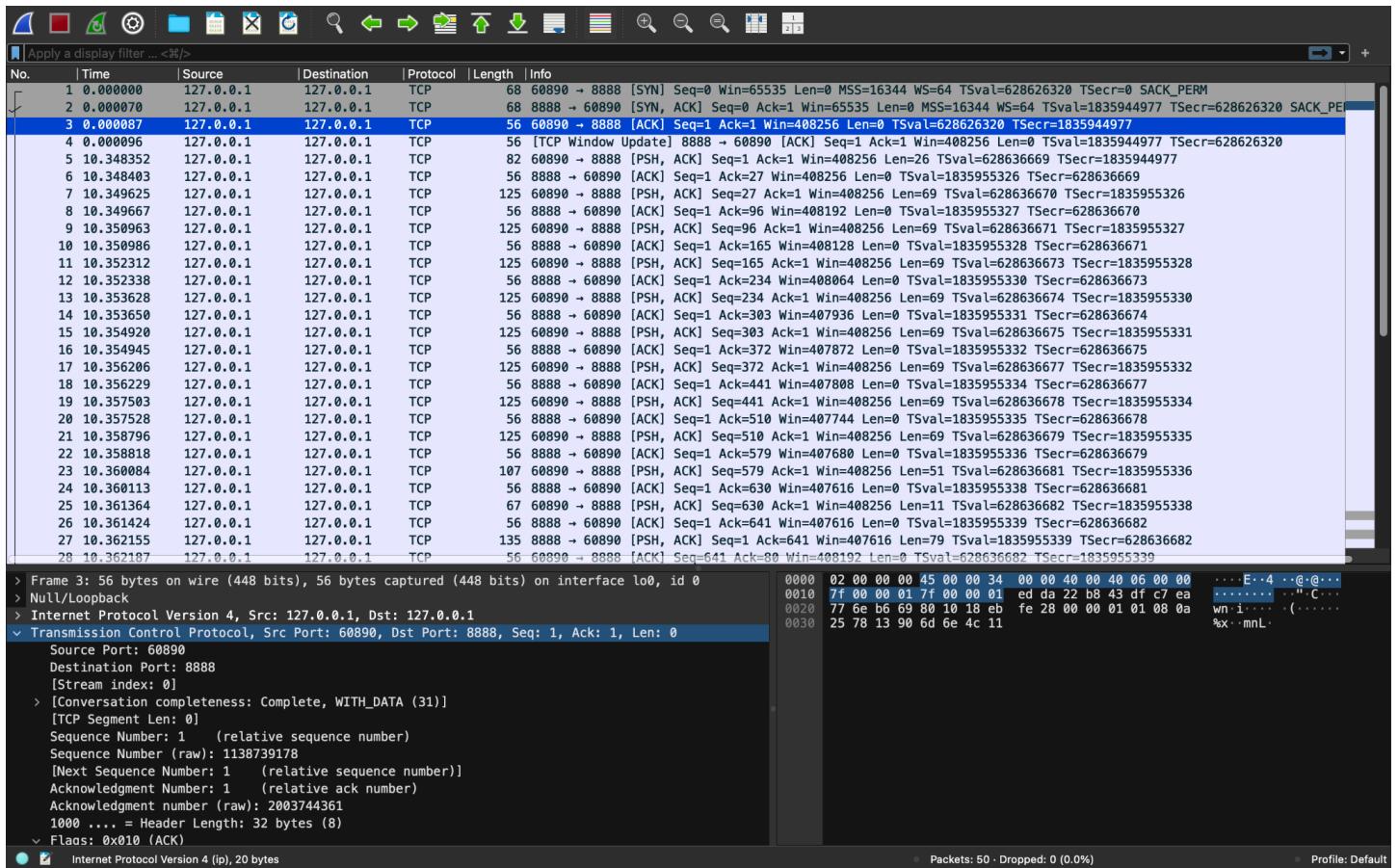
1. What are the source and destination IP addresses and ports? Share the screenshots to justify your answer.



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



### 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.

No.	Time	Source	Destination	Protocol	Length	Info
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
24	10.360113	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	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
26	10.361424	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	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
28	10.362187	127.0.0.1	127.0.0.1	TCP	56	60890 → 8888 [ACK] Seq=641 Ack=80 Win=408192 Len=0 TSval=628636682 TSecr=1835955339
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
30	10.363356	127.0.0.1	127.0.0.1	TCP	56	60890 → 8888 [ACK] Seq=641 Ack=159 Win=408128 Len=0 TSval=628636684 TSecr=1835955341
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=628636684
32	10.364508	127.0.0.1	127.0.0.1	TCP	56	60890 → 8888 [ACK] Seq=641 Ack=238 Win=408000 Len=0 TSval=628636685 TSecr=1835955342
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
34	10.365639	127.0.0.1	127.0.0.1	TCP	56	60890 → 8888 [ACK] Seq=641 Ack=317 Win=407936 Len=0 TSval=628636686 TSecr=1835955343
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
36	10.366794	127.0.0.1	127.0.0.1	TCP	56	60890 → 8888 [ACK] Seq=641 Ack=396 Win=407872 Len=0 TSval=628636687 TSecr=1835955344
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
38	10.367937	127.0.0.1	127.0.0.1	TCP	56	60890 → 8888 [ACK] Seq=641 Ack=475 Win=407808 Len=0 TSval=628636688 TSecr=1835955345
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
40	10.369086	127.0.0.1	127.0.0.1	TCP	56	60890 → 8888 [ACK] Seq=641 Ack=554 Win=407744 Len=0 TSval=628636690 TSecr=1835955347
41	10.370232	127.0.0.1	127.0.0.1	TCP	106	8888 → 60890 [PSH, ACK] Seq=554 Ack=641 Win=407616 Len=50 TSval=1835955348 TSecr=628636690
42	10.370254	127.0.0.1	127.0.0.1	TCP	56	60890 → 8888 [ACK] Seq=641 Ack=684 Win=407680 Len=0 TSval=628636691 TSecr=1835955348
43	10.371391	127.0.0.1	127.0.0.1	TCP	67	8888 → 60890 [PSH, ACK] Seq=684 Ack=641 Win=407616 Len=11 TSval=1835955349 TSecr=628636691
44	10.371422	127.0.0.1	127.0.0.1	TCP	56	60890 → 8888 [ACK] Seq=641 Ack=615 Win=407680 Len=0 TSval=628636692 TSecr=1835955349
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
46	12.630268	127.0.0.1	127.0.0.1	TCP	56	8888 → 60890 [ACK] Seq=615 Ack=643 Win=407616 Len=0 TSval=1835957608 TSecr=628638951
47	12.630370	127.0.0.1	127.0.0.1	TCP	56	60890 → 8888 [FIN, ACK] Seq=643 Ack=615 Win=407680 Len=0 TSval=628638951 TSecr=1835957608
48	12.630396	127.0.0.1	127.0.0.1	TCP	56	8888 → 60890 [ACK] Seq=615 Ack=644 Win=407616 Len=0 TSval=1835957608 TSecr=628638951
49	12.630407	127.0.0.1	127.0.0.1	TCP	56	8888 → 60890 [FIN, ACK] Seq=615 Ack=644 Win=407616 Len=0 TSval=1835957608 TSecr=628638951
50	12.630462	127.0.0.1	127.0.0.1	TCP	56	60890 → 8888 [ACK] Seq=644 Ack=616 Win=407680 Len=0 TSval=628638951 TSecr=1835957608

We can find the sequence at the end:

- FIN (Client → Server)
- ACK (Server → Client)
- FIN (Server → Client)
- ACK (Client → Server)

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 TStamp=628636669 TSect=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 TStamp=628636670 TSect=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 TStamp=628636671 TSect=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 TStamp=628636673 TSect=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 TStamp=628636674 TSect=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 TStamp=628636675 TSect=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 TStamp=628636677 TSect=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 TStamp=628636678 TSect=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 TStamp=628636679 TSect=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 TStamp=628636681 TSect=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 TStamp=628636682 TSect=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 TStamp=1835955339 TSect=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 TStamp=1835955341 TSect=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 TStamp=1835955342 TSect=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 TStamp=1835955343 TSect=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 TStamp=1835955344 TSect=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 TStamp=1835955345 TSect=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 TStamp=1835955347 TSect=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 TStamp=1835955348 TSect=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 TStamp=1835955349 TSect=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 TStamp=628638951 TSect=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 DEFPRTWV
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 !3 ?1 > ls 04:22:41 pm
CS39006_Networks_Lab_Assignment3.pdf readme.md
TCP_packets.pcapng retrieveencfileclient.c
doencfileserver.c sample.txt
key.txt sample2.txt
makefile tut
~/Doc/6/CN/LAB/LA3 main !3 ?1 > cat sample2.txt 04:22:44 pm
A sockfd 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 sockfd. This descriptor is then used with other functions like bind, listen and accept to set up the server and manage incoming connections

The sockfd 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 !3 ?1 > make runsv 04:22:49 pm
gcc -Wall -o svr doencfileserver.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 !3 ?1 > make runcl 04:22:49 pm
gcc -Wall -o cli retrieveencfileclient.c
./cli

Enter filename to encrypt: sample2.txt
Enter 26-character encryption key: DEFPRTVWLMZAYGHQSIUJXKBCN0
[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 !3 ?6 > cat sample2.txt.enc 13s 04:23:28 pm
D uhfztp tlar pruflqjhi lu dg lgjrvri jwdj xglsxran lprqjltlu d uhfzrj bljwlg d qihfru
u Jwlw pruflqjhi lu xurp jh qrithiy hqridjlhg u hg jwr uhfzrj uxfw du rldplvg tihi bi
ljgvj jh lj Bwrg firdjlgv d urikri jwr uhfzrj txgfjlhg lu fdaarp jh lglijdalar d uhfztp
Jwlw pruflqjhi lu jwrg xurp bljw hjwri txgfjlhg alzr elgp alujrg dgp dffrqj jh urj xq
jwru urikri dgp ydgdr lgfhylvg fhggrfjlg u

Jwr uhfztp lu fixflda thi ydgdlvg fhyyxglfdjlg lkri jwr grjbhiz dgp lu jirdjrp ulyladi
an jh hjwri tlar pruflqjhiu lg xglic alzr unujryu daahblvg hqridjlhg u alzr irdp dgp bilj
r jh er xurp hg uhfzrju du braa
~/Doc/6/CN/LAB/LA3 main !3 ?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 urikri dgp ydgdrv lgfhylgv fhggrfjlhg
Jwr uhfztp lu fixflda thi ydgdlvg fhyyxglfdjlg hkri jwr grjbhiz dgp lu jirdjrp ulyldi
an jh hjiwr tlar poflqjhui lg Xglic alzr unujryu daahblgv hqridjlgwu alzr irdp dgp bilj
r jh er xurp ho uhfzriu du braa
~/Doc/6/CN/LAB/LA3 main !3 ?6 > [ 04:23:52 pm

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

Enter filename to encrypt: sample3.txt
Enter 26-character encryption key: DEFPRTVWLMZAYGHQSIUJXKBCN0
[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: DEFPRTVWLMZAYGHQSIUJXKBCN0
[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 !3 ?11 > [ 04:41:12 pm
40s 04:41:12 pm
```

From the above three sample files:

1) 603 bytes -> 17 packets

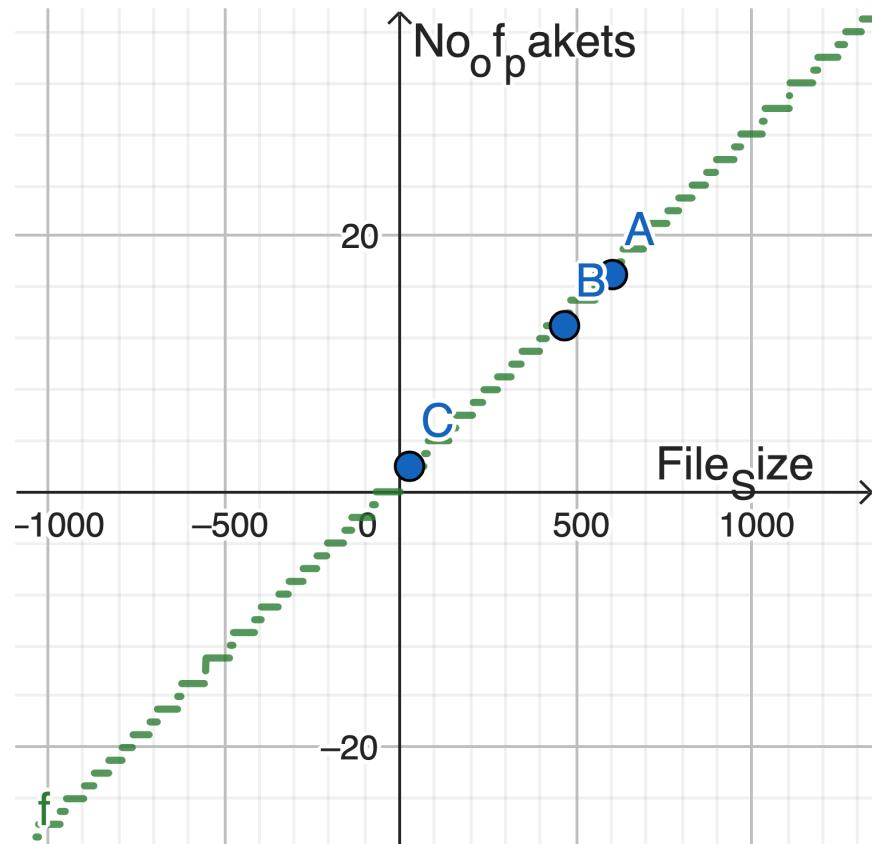
2) 467 bytes -> 13 packets

3) 27 bytes -> 2 packets

Mathematically: for a file of size  $B$  bytes

→ Client will send  $\lceil B/69 \rceil$  packets

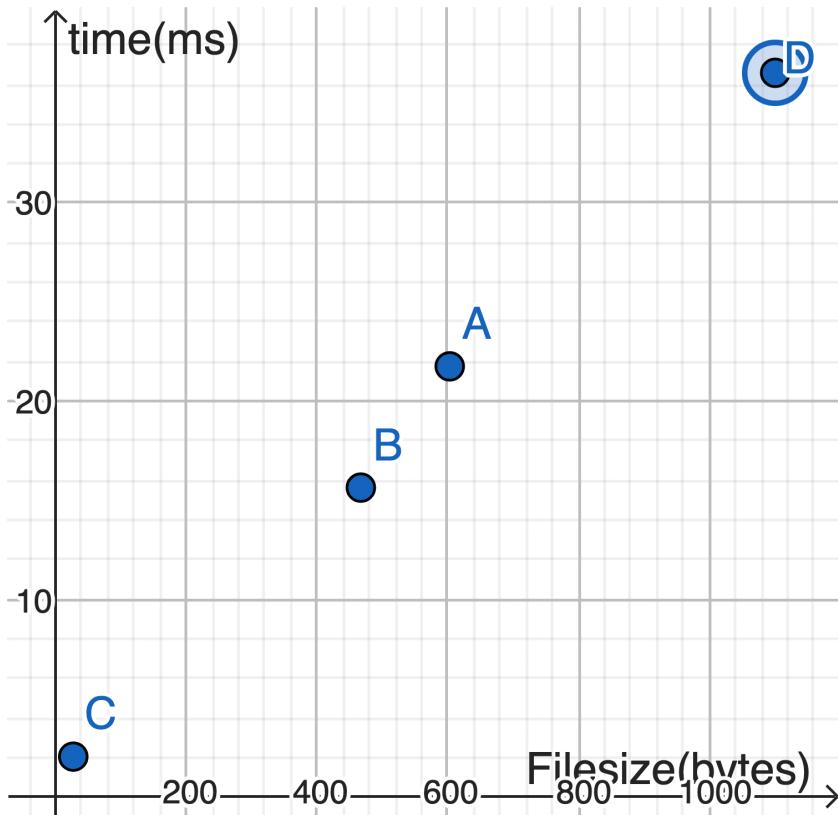
→ Server will send  $\lceil B/79 \rceil$  packets



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.

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	10.354928	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	10.357583	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=628636684
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=50 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

- Total Time taken for the file transfer of size 603 bytes =  $10.371391 - 10.349626 = 0.0217652$  seconds
- More observations
  - 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



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

Packet Lengths:									
Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start	
Packet Lengths	50	81.60	56	135	0.0040	100%	0.4000	10.348	
0-19	0	-	-	-	0.0000	0.00%	-	-	
20-39	0	-	-	-	0.0000	0.00%	-	-	
40-79	32	57.50	56	68	0.0025	64.00%	0.2200	10.348	
80-159	18	124.44	82	135	0.0014	36.00%	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%	-	-	

Using the Statistics button in wireshark, the **average packet length is 81.60 bytes** for the file size 603 bytes.