Assignment- 2

Problem Statement:

The objective of this laboratory exercise is to look at the details of the Transmission Control Protocol (TCP). TCP is a transport layer protocol. It is used by many application protocols like HTTP, FTP, SSH etc., where guaranteed and reliable delivery of messages is required. To do this exercise you need to install the Wireshark tool. This tool would be used to capture and examine a packet trace. Wireshark can be downloaded from www.wireshark.org.

Step1: Capture a Trace

1. Launch Wireshark
2. From Capture→Options select Loopback interface
3. Start a capture with a filter of "ip.addr==127.0.0.1 and tcp.port==xxxx", where xxxx is the port number used by the TCP server.
4. Run the TCP server program on a terminal.
5. Run two instances of the TCP client program on two separate terminals and send some dummy data to the sever.
6. Stop Wireshark capture

Step2: TCP Connection Establishment

To observe the three-way handshake in action, look for a TCP segment with SYN flag set. A "SYN" segment is the start of the three-way handshake and is sent by the TCP client to the TCP server. The server then replies with a TCP segment with SYN and ACK flag set. And finally the client sends an "ACK" to the server. For all the above three segments record the values of the sequence number and acknowledgment fields. Draw a time sequence diagram of the three-way handshake for TCP connection establishment in your trace. Do it for all the client connections.

Step3: TCP Data Transfer

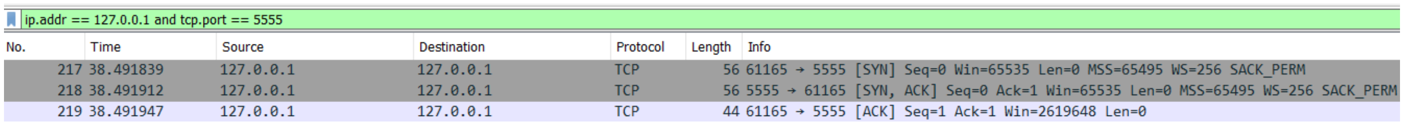
For all data segments sent by the client, record the value of the sequence number and acknowledge number fields. Also, record the same for the corresponding acknowledgements sent by the server. Draw a time sequence diagram of the data transfer in your trace. Do it for all the client connections.

Step4: TCP Connection Termination

Once the data transfer is over, the client initiates the connection termination by sending TCP segment with FIN flag set, to the server. Server acknowledges it and sends it's own intention to terminate the connection by sending a TCP segment with FIN and ACK flags set. The client finally sends an ACK segment to the server. For all the above three segments record the values of the sequence number and acknowledgment fields. Draw a time sequence diagram of the three-way handshake for TCP connection termination in your trace. Do it for all the client connections.

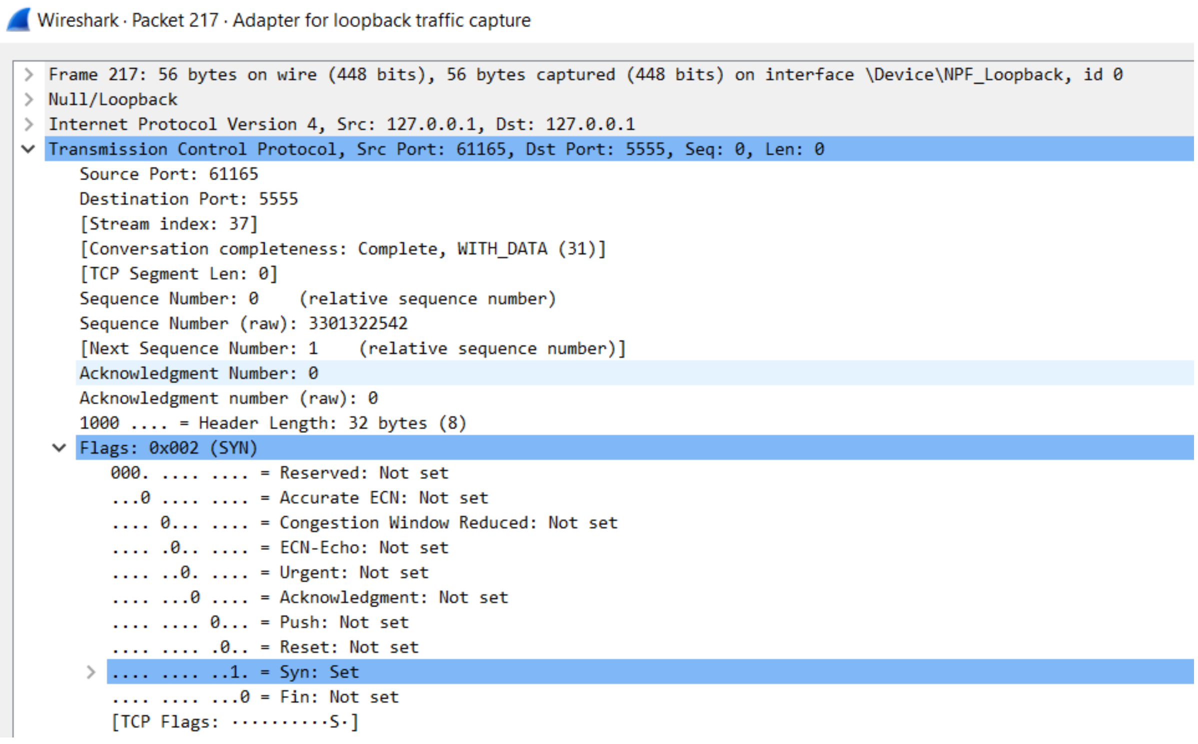
TCP connection establishment:

A snapshot of the 3-way handshake for TCP connection establishment.



```
ip.addr == 127.0.0.1 and tcp.port == 5555
No.     Time          Source        Destination   Protocol  Length  Info
    217 38.491839     127.0.0.1     127.0.0.1     TCP       56 61165 → 5555 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
    218 38.491912     127.0.0.1     127.0.0.1     TCP       56 5555 → 61165 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
    219 38.491947     127.0.0.1     127.0.0.1     TCP       44 61165 → 5555 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
```

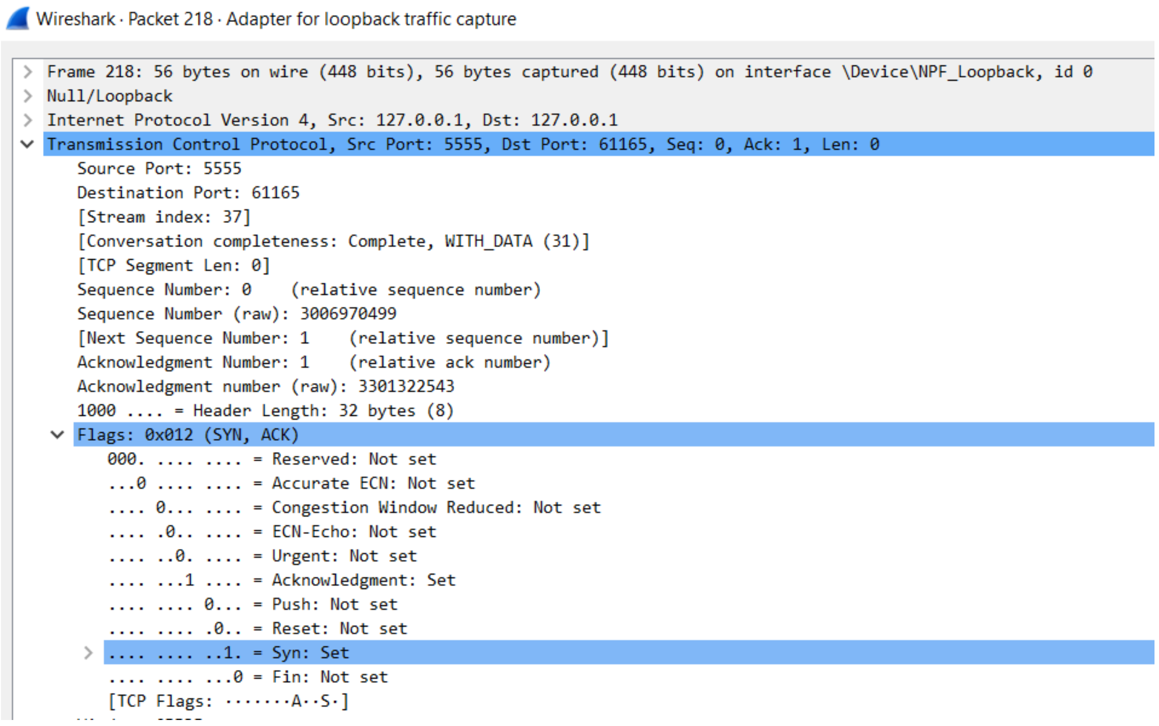
Step-1: Client sends SYN (Synchronize) message to the server. The message includes SYN flag set to 1 and it contains a unique sequence number which is any 32-bit number and acknowledge number which is set to 0.



```
Wireshark · Packet 217 · Adapter for loopback traffic capture

> Frame 217: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
v Transmission Control Protocol, Src Port: 61165, Dst Port: 5555, Seq: 0, Len: 0
     Source Port: 61165
     Destination Port: 5555
     [Stream index: 37]
     [Conversation completeness: Complete, WITH_DATA (31)]
     [TCP Segment Len: 0]
     Sequence Number: 0    (relative sequence number)
     Sequence Number (raw): 3301322542
     [Next Sequence Number: 1    (relative sequence number)]
     Acknowledgment Number: 0
     Acknowledgment number (raw): 0
     1000 .... = Header Length: 32 bytes (8)
   v 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·]
```

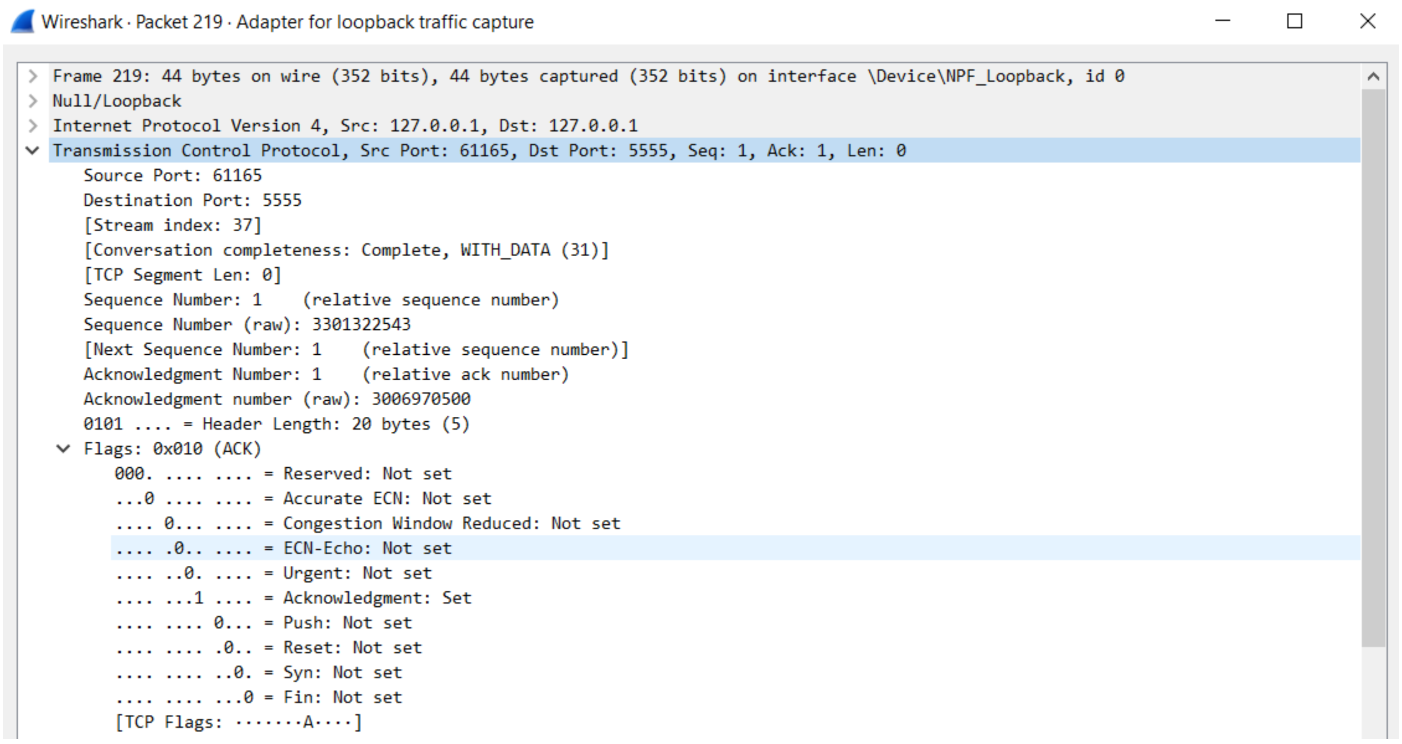
Step-2: Now the server responds with SYN and ACK (Acknowledge) message to the client. Here the SYN and ACK flags are set to 1. The ACK number is one higher than the SYN sequence number received by the client. But the sequence number will be different.



```
Wireshark · Packet 218 · Adapter for loopback traffic capture

> Frame 218: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
v Transmission Control Protocol, Src Port: 5555, Dst Port: 61165, Seq: 0, Ack: 1, Len: 0
     Source Port: 5555
     Destination Port: 61165
     [Stream index: 37]
     [Conversation completeness: Complete, WITH_DATA (31)]
     [TCP Segment Len: 0]
     Sequence Number: 0    (relative sequence number)
     Sequence Number (raw): 3006970499
     [Next Sequence Number: 1    (relative sequence number)]
     Acknowledgment Number: 1    (relative ack number)
     Acknowledgment number (raw): 3301322543
     1000 .... = Header Length: 32 bytes (8)
   v Flags: 0x012 (SYN, ACK)
        000. .... .... = Reserved: Not set
        ...0 .... .... = Accurate ECN: Not set
        .... 0... .... = Congestion Window Reduced: Not set
        .... .0.. .... = ECN-Echo: Not set
        .... ..0. .... = Urgent: Not set
        .... ...1 .... = Acknowledgment: Set
        .... .... 0... = Push: Not set
        .... .... .0.. = Reset: Not set
      > .... .... ..1. = Syn: Set
        .... .... ...0 = Fin: Not set
        [TCP Flags: ·······A··S·]
```

Step-3: After client has received the SYN – ACK message from server it sends and Acknowledge message to the server. The ACK flag is set to 1 and the ACK number will be one higher than the previously received sequence number from server.

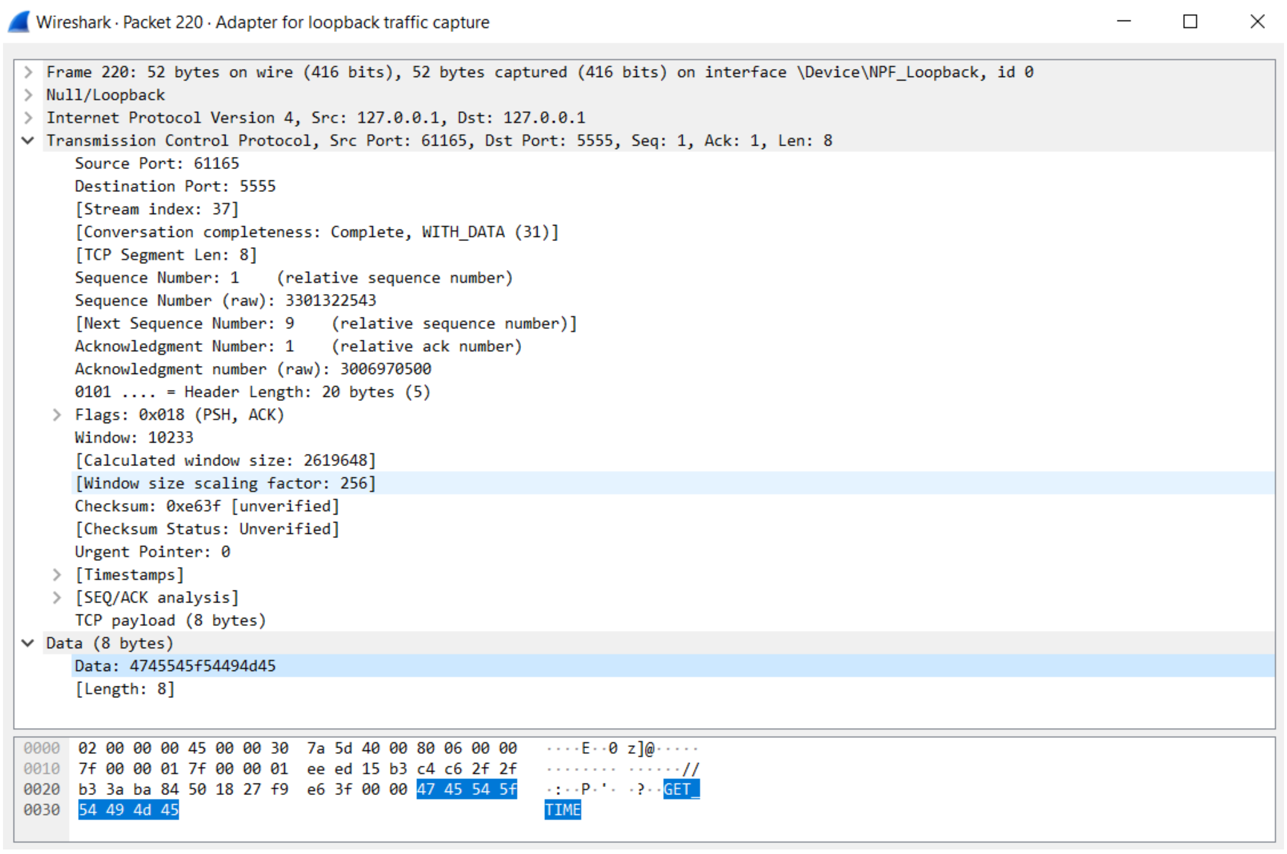


Wireshark · Packet 219 · Adapter for loopback traffic capture

```
> Frame 219: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
v Transmission Control Protocol, Src Port: 61165, Dst Port: 5555, Seq: 1, Ack: 1, Len: 0
     Source Port: 61165
     Destination Port: 5555
     [Stream index: 37]
     [Conversation completeness: Complete, WITH_DATA (31)]
     [TCP Segment Len: 0]
     Sequence Number: 1    (relative sequence number)
     Sequence Number (raw): 3301322543
     [Next Sequence Number: 1    (relative sequence number)]
     Acknowledgment Number: 1    (relative ack number)
     Acknowledgment number (raw): 3006970500
     0101 .... = Header Length: 20 bytes (5)
   v Flags: 0x010 (ACK)
        000. .... .... = Reserved: Not set
        ...0 .... .... = Accurate ECN: Not set
        .... 0... .... = Congestion Window Reduced: Not set
        .... .0.. .... = ECN-Echo: Not set
        .... ..0. .... = Urgent: Not set
        .... ...1 .... = Acknowledgment: Set
        .... .... 0... = Push: Not set
        .... .... .0.. = Reset: Not set
        .... .... ..0. = Syn: Not set
        .... .... ...0 = Fin: Not set
        [TCP Flags: ·······A····]
```

TCP Data Transfer:

After successful connection data can be transferred in between the server and the client. Whoever wants to send data will send a message with flags PSH and ACK set to 1 along with the actual data.

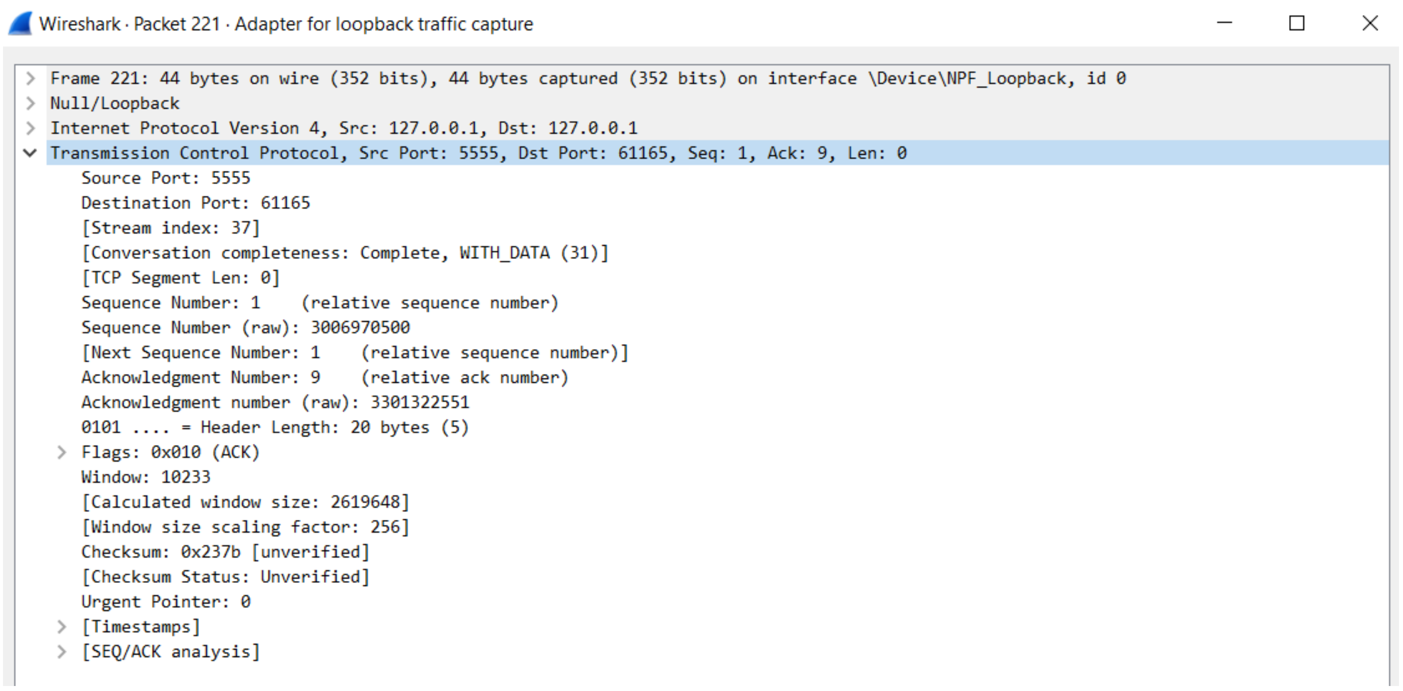


Wireshark · Packet 220 · Adapter for loopback traffic capture

```
> Frame 220: 52 bytes on wire (416 bits), 52 bytes captured (416 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
v Transmission Control Protocol, Src Port: 61165, Dst Port: 5555, Seq: 1, Ack: 1, Len: 8
     Source Port: 61165
     Destination Port: 5555
     [Stream index: 37]
     [Conversation completeness: Complete, WITH_DATA (31)]
     [TCP Segment Len: 8]
     Sequence Number: 1    (relative sequence number)
     Sequence Number (raw): 3301322543
     [Next Sequence Number: 9    (relative sequence number)]
     Acknowledgment Number: 1    (relative ack number)
     Acknowledgment number (raw): 3006970500
     0101 .... = Header Length: 20 bytes (5)
   > Flags: 0x018 (PSH, ACK)
     Window: 10233
     [Calculated window size: 2619648]
     [Window size scaling factor: 256]
     Checksum: 0xe63f [unverified]
     [Checksum Status: Unverified]
     Urgent Pointer: 0
   > [Timestamps]
   > [SEQ/ACK analysis]
     TCP payload (8 bytes)
v Data (8 bytes)
     Data: 4745545f54494d45
     [Length: 8]

0000  02 00 00 00 45 00 00 30  7a 5d 40 00 80 06 00 00   ····E··0 z]@·····
0010  7f 00 00 01 7f 00 00 01  ee ed 15 b3 c4 c6 2f 2f   ········ ······//
0020  b3 3a ba 84 50 18 27 f9  e6 3f 00 00 47 45 54 5f   ·:··P·'· ·?··GET_
0030  54 49 4d 45                                        TIME
```

When the other computer receives the data, it will send another ACK message to the sender. In this case the ACK number will be sequence number of previous message + number of bytes sent.



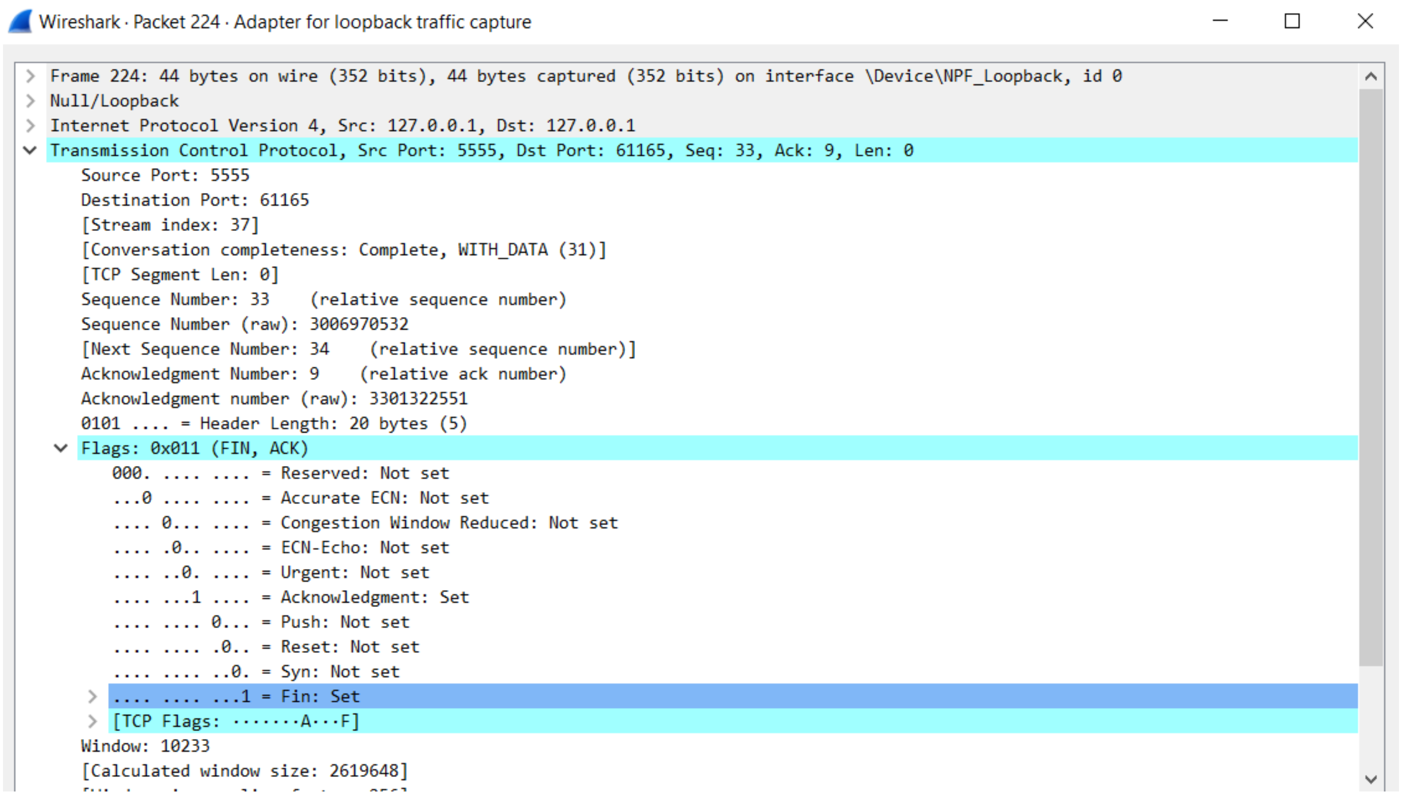
Wireshark · Packet 221 · Adapter for loopback traffic capture

```
> Frame 221: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
v Transmission Control Protocol, Src Port: 5555, Dst Port: 61165, Seq: 1, Ack: 9, Len: 0
    Source Port: 5555
    Destination Port: 61165
    [Stream index: 37]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 3006970500
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 9    (relative ack number)
    Acknowledgment number (raw): 3301322551
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window: 10233
    [Calculated window size: 2619648]
    [Window size scaling factor: 256]
    Checksum: 0x237b [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > [Timestamps]
  > [SEQ/ACK analysis]
```

TCP Connection Termination:

For connection termination one computer say the server will send a message with FIN and ACK flag set to 1. The client upon receiving the message will send an ACK message to the previous sender as acknowledge for closing the connection.

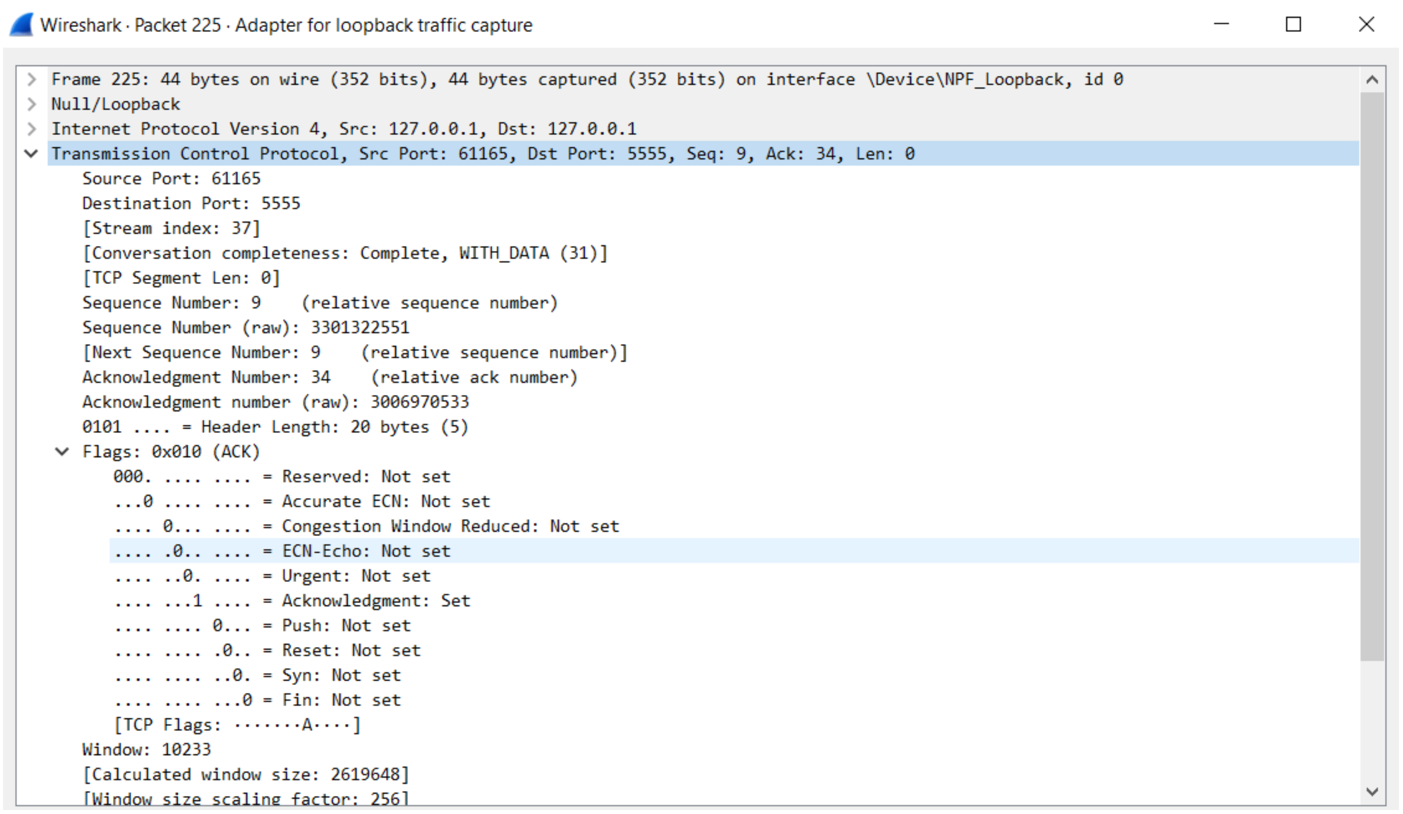
Similarly, the client will now send a message with FIN and ACK set to 1 to server and server will send the acknowledge as FIN, ACK back to the client.



Wireshark · Packet 224 · Adapter for loopback traffic capture

```
> Frame 224: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
v Transmission Control Protocol, Src Port: 5555, Dst Port: 61165, Seq: 33, Ack: 9, Len: 0
    Source Port: 5555
    Destination Port: 61165
    [Stream index: 37]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 33    (relative sequence number)
    Sequence Number (raw): 3006970532
    [Next Sequence Number: 34    (relative sequence number)]
    Acknowledgment Number: 9    (relative ack number)
    Acknowledgment number (raw): 3301322551
    0101 .... = Header Length: 20 bytes (5)
  v Flags: 0x011 (FIN, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Accurate ECN: Not set
      .... 0... .... = Congestion Window Reduced: Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
    > .... .... ...1 = Fin: Set
    > [TCP Flags: ·······A···F]
    Window: 10233
    [Calculated window size: 2619648]
```

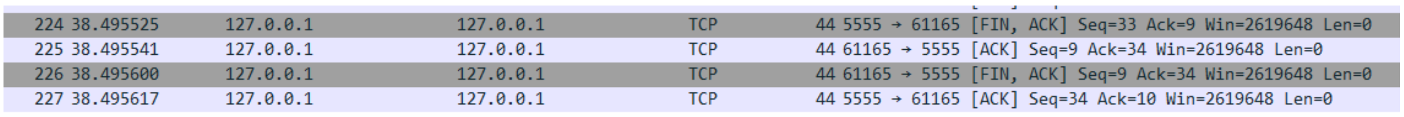
For the ACK message the ACK number will be one higher than the sequence number of the previous (FIN, ACK) message.



Wireshark · Packet 225 · Adapter for loopback traffic capture

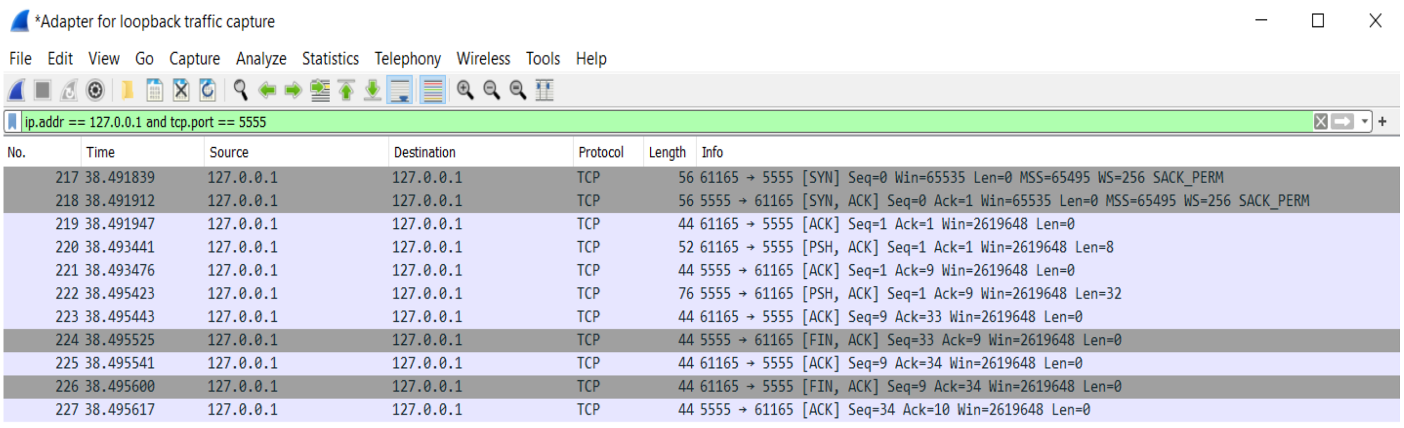
```
> Frame 225: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
v Transmission Control Protocol, Src Port: 61165, Dst Port: 5555, Seq: 9, Ack: 34, Len: 0
    Source Port: 61165
    Destination Port: 5555
    [Stream index: 37]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 9    (relative sequence number)
    Sequence Number (raw): 3301322551
    [Next Sequence Number: 9    (relative sequence number)]
    Acknowledgment Number: 34    (relative ack number)
    Acknowledgment number (raw): 3006970533
    0101 .... = Header Length: 20 bytes (5)
  v Flags: 0x010 (ACK)
        000. .... .... = Reserved: Not set
        ...0 .... .... = Accurate ECN: Not set
        .... 0... .... = Congestion Window Reduced: Not set
        .... .0.. .... = ECN-Echo: Not set
        .... ..0. .... = Urgent: Not set
        .... ...1 .... = Acknowledgment: Set
        .... .... 0... = Push: Not set
        .... .... .0.. = Reset: Not set
        .... .... ..0. = Syn: Not set
        .... .... ...0 = Fin: Not set
        [TCP Flags: ·······A····]
    Window: 10233
    [Calculated window size: 2619648]
    [Window size scaling factor: 256]
```

This is called a 4-way handshake, which is required for closing a TCP connection. This is actually a set of 2-way handshake.



```
224 38.495525   127.0.0.1   127.0.0.1   TCP   44 5555 → 61165 [FIN, ACK] Seq=33 Ack=9 Win=2619648 Len=0
225 38.495541   127.0.0.1   127.0.0.1   TCP   44 61165 → 5555 [ACK] Seq=9 Ack=34 Win=2619648 Len=0
226 38.495600   127.0.0.1   127.0.0.1   TCP   44 61165 → 5555 [FIN, ACK] Seq=9 Ack=34 Win=2619648 Len=0
227 38.495617   127.0.0.1   127.0.0.1   TCP   44 5555 → 61165 [ACK] Seq=34 Ack=10 Win=2619648 Len=0
```

<u>Snapshot of everything altogether:</u>

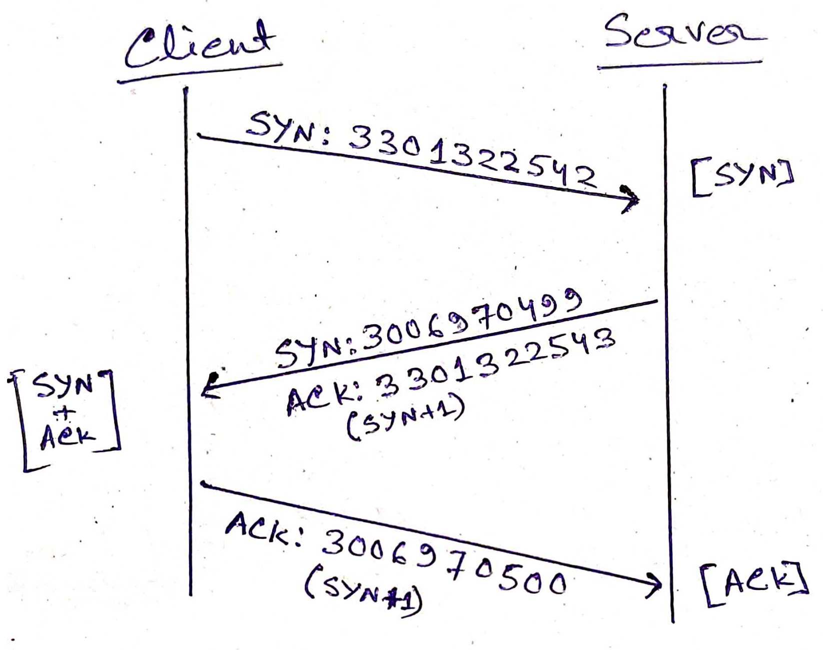

*Adapter for loopback traffic capture
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr == 127.0.0.1 and tcp.port == 5555

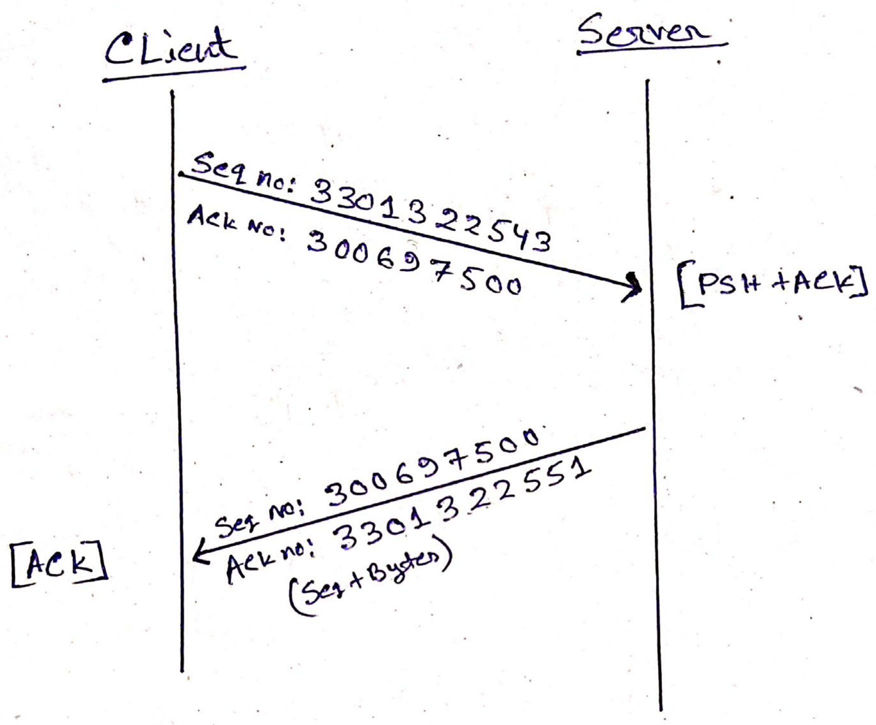
```
No.  Time        Source      Destination  Protocol  Length  Info
217 38.491839   127.0.0.1   127.0.0.1    TCP       56 61165 → 5555 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
218 38.491912   127.0.0.1   127.0.0.1    TCP       56 5555 → 61165 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
219 38.491947   127.0.0.1   127.0.0.1    TCP       44 61165 → 5555 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
220 38.493441   127.0.0.1   127.0.0.1    TCP       52 61165 → 5555 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=8
221 38.493476   127.0.0.1   127.0.0.1    TCP       44 5555 → 61165 [ACK] Seq=1 Ack=9 Win=2619648 Len=0
222 38.495423   127.0.0.1   127.0.0.1    TCP       76 5555 → 61165 [PSH, ACK] Seq=1 Ack=9 Win=2619648 Len=32
223 38.495443   127.0.0.1   127.0.0.1    TCP       44 61165 → 5555 [ACK] Seq=9 Ack=33 Win=2619648 Len=0
224 38.495525   127.0.0.1   127.0.0.1    TCP       44 5555 → 61165 [FIN, ACK] Seq=33 Ack=9 Win=2619648 Len=0
225 38.495541   127.0.0.1   127.0.0.1    TCP       44 61165 → 5555 [ACK] Seq=9 Ack=34 Win=2619648 Len=0
226 38.495600   127.0.0.1   127.0.0.1    TCP       44 61165 → 5555 [FIN, ACK] Seq=9 Ack=34 Win=2619648 Len=0
227 38.495617   127.0.0.1   127.0.0.1    TCP       44 5555 → 61165 [ACK] Seq=34 Ack=10 Win=2619648 Len=0
```

Timing sequence diagram:

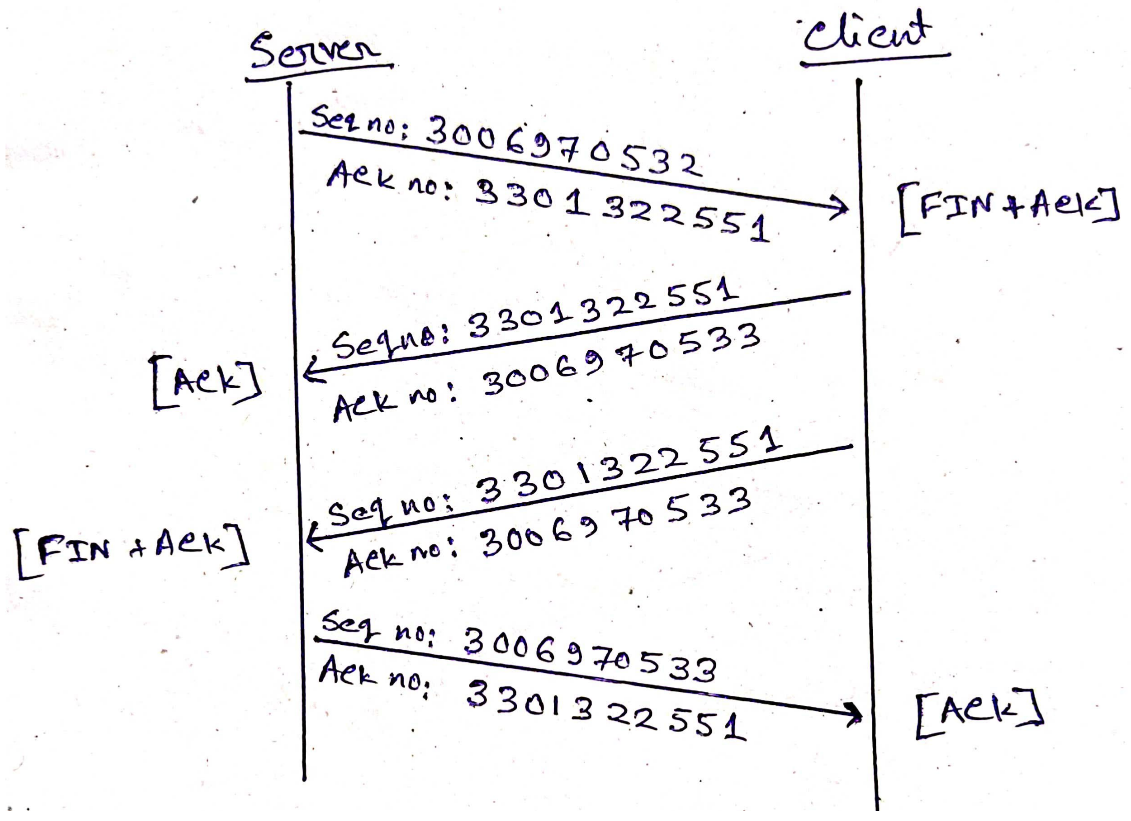
TCP connection:



Data transfer:



Closing connection:



Server
Client
Seq no: 3006970532
Ack no: 3301322551
[FIN + Ack]
Seq no: 3301322551
Ack no: 3006970533
[Ack]
Seq no: 3301322551
Ack no: 3006970533
[FIN + Ack]
Seq no: 3006970533
Ack no: 3301322551
[Ack]