



Student Details:

Name: AB Satyaprakash

Roll Number: 180123062

Department: Mathematics and Computing

Question 1.

The application assigned to me was **FortiClient VPN (Desktop App)**. The protocols used by the application at the different layers are :
(the physical layer protocols couldn't be figured out from the traces)

1. Application Layer :

- SSH (Secure Shell Protocol)
- TLSv1.2 (Transport Layer Security Protocol)
- DNS (Domain Network System Protocol)

2. Transport Layer :

- TCP (Transmission Control Protocol)
- UDP (User Datagram Protocol)

3. Network Layer :

- IPv4 (Internet Protocol Version 4)

4. Link Layer :

- Ethernet II

1. Application Layer

a. The SSH packet format has the following fields:

- **Packet Length** is the length of the packet in bytes, not including **HMAC** value or the **Packet Length** field itself.
- **Padding Length** is the length of random padding in bytes.
- **Payload** is the useful contents of the packet. If compression has been negotiated, this field is compressed.
- **Random Padding** is the arbitrary-length padding, such that the total length of (packet_length || padding_length || payload || random padding) is a multiple of the cipher block size or 8, whichever is larger.
- **MAC (Message Authentication Code)** : MAC (Message Authentication Code) field contains the MAC value, if the message authentication has been negotiated, this field contains the MAC bytes.

```
Frame 4: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface ppp0, id 0
Linux cooked capture
Internet Protocol Version 4, Src: 172.16.70.72 (172.16.70.72), Dst: 172.16.18.18 (172.16.18.18)
Transmission Control Protocol, Src Port: ssh (22), Dst Port: 50254 (50254), Seq: 1, Ack: 37, Len: 36
SSH Protocol
  Packet Length (encrypted): d3f972b5
  Encrypted Packet: d47bd0c6146f2cea1b9fa02a952a0574502d1ddc9241b4ad...
  [Direction: server-to-client]
```

b. The TLS (v1.2 or v1.3) packet format has the following fields:

- **Content type** represents the type of data being encrypted
- **Version** represents the version of TLS (in my case TLSv1.2)
- **Length** represents the length of the packet in bytes
- **Encrypted Application Data** is the data that has been encrypted by the TLS

```
Frame 20: 111 bytes on wire (888 bits), 111 bytes captured (888 bits) on interface wlo1, id 0
Ethernet II, Src: AzureWav_e7:aa:8b (80:c5:f2:e7:aa:8b), Dst: d2:f8:8c:0b:8b:ab (d2:f8:8c:0b:8b:ab)
Internet Protocol Version 4, Src: 172.16.18.18 (172.16.18.18), Dst: agnigarh.iitg.ac.in (14.139.196.11)
Transmission Control Protocol, Src Port: 46710 (46710), Dst Port: cirrossp (10443), Seq: 1, Ack: 1, Len: 45
Transport Layer Security
  TLSv1.2 Record Layer: Application Data Protocol: Application Data
    Content Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 40
    Encrypted Application Data: bfba16d455b949a74838b078e49361317d31198bea41110f...
```

c. The DNS packet has the following fields:

- **Transaction ID** is a 16 bit header section in a DNS message
- **DNS Header Flag** is a flag field in the 2nd 16 bit word of the query
- **Questions** represent the number of questions asked in the message in a specified format.
- **Answer RRs, Authority RRs and Additional RRs** are the various resource records.
- **Queries** represent the domain that is being queried.
- **Type** is the type of resource record
- **Class** represents class code.

```
Frame 21: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface ppp0, id 0
Linux cooked capture
User Datagram Protocol, Src Port: 51443 (51443), Dst Port: domain (53)
Domain Name System (query)
  Transaction ID: 0xa6fe
  Flags: 0x0100 Standard query
    0... .. = Response: Message is a query
    .000 0... .. = Opcode: Standard query (0)
    .... 0... .. = Truncated: Message is not truncated
    .... 1... .. = Recursion desired: Do query recursively
    .... 0... .. = Z: reserved (0)
    .... 0... .. = Non-authenticated data: Unacceptable
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 1
  Queries
    location.services.mozilla.com: type A, class IN
  Additional records
  [Response In: 24]
```

2. Transport Layer

a. The TCP packet format has the following fields:

- **Source Port and Destination Port fields** (16 bits each) identify the end points of the connection.
- **Sequence Number field** (32 bits) specifies the number assigned to the first byte of data in the current message.
- **Acknowledgement Number field** (32 bits) contains the value of the next sequence number that the sender of the segment is expecting to receive, if the ACK control bit is set.
- **Data Offset (a.k.a. Header Length) field** (variable length) tells how many 32-bit words are contained in the TCP header.
- **Reserved field** (6 bits) must be zero. This is for future use.
- **Flags field** (6 bits) contains the various flags:
 - URG—Indicates that some urgent data has been placed.
 - ACK—Indicates that acknowledgement number is valid.
 - PSH—Indicates that data should be passed to the application as soon as possible.
 - RST—Resets the connection.
 - SYN—Synchronizes sequence numbers to initiate a connection.
 - FIN—Means that the sender of the flag has finished sending data.
- **Window field** (16 bits) specifies the size of the sender's receive window (that is, buffer space available for incoming data).
- **Checksum field** (16 bits) indicates whether the header was damaged in transit.
- **Urgent pointer field** (16 bits) points to the first urgent data byte in the packet.
- **Options field** (variable length) specifies various TCP options.
- **Data field** (variable length) contains upper-layer information.

```
Frame 21: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface wlo1, id 0
  Ethernet II, Src: d2:f8:8c:0b:8b:ab (d2:f8:8c:0b:8b:ab), Dst: AzureWav_e7:aa:8b (88:c5:f2:e7:aa:8b)
  Internet Protocol Version 4, Src: agnigarh.iitg.ac.in (14.139.196.11), Dst: 192.168.43.225 (192.168.43.225)
  Transmission Control Protocol, Src Port: cirrossp (10443), Dst Port: 46710 (46710), Seq: 1, Ack: 46, Len: 0
    Source Port: cirrossp (10443)
    Destination Port: 46710 (46710)
    [Stream index: 4]
    [TCP Segment Len: 0]
    Sequence number: 1 (relative sequence number)
    Sequence number (raw): 31531984
    [Next sequence number: 1 (relative sequence number)]
    Acknowledgment number: 46 (relative ack number)
    Acknowledgment number (raw): 293787911
    1000 .... = Header Length: 32 bytes (8)
    Flags: 0x010 (ACK)
    Window size value: 162
    [Calculated window size: 162]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x0df1 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
    Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    [SEQ/ACK analysis]
    [Timestamps]
```

b. The UDP packet format has the following fields:

- **Source port number** identifies the sender's port, when used, and should be assumed to be the port to reply to if needed.
- **Destination port number** identifies the receiver's port and is required.
- **Length** specifies the length in bytes of the UDP header and UDP data.
- **Checksum** field may be used for error-checking of the header and data.

```
Frame 21: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface ppp0, id 0
  Linux cooked capture
  Internet Protocol Version 4, Src: 172.18.16.18 (172.18.16.18), Dst: 172.17.1.1 (172.17.1.1)
  User Datagram Protocol, Src Port: 51443 (51443), Dst Port: domain (53)
    Source Port: 51443 (51443)
    Destination Port: domain (53)
    Length: 66
    Checksum: 0xf93c [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
    [Timestamps]
  Domain Name System (query)
```

3. Network Layer:

a. The IPv4 packet has the following fields:

- **Version** is the version number of Internet Protocol used (e.g. IPv4).
- **IHL** is the length of the entire IP header.
- **DSCP** is Differentiated Services Code Point. This is a type of service.
- **ECN** is Explicit Congestion Notification. It carries information about the congestion seen in the route.
- **Total Length** is the length of the entire IP Packet (including IP header and IP Payload).

```
Frame 23: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface wlo1, id 0
  Ethernet II, Src: AzureWav_e7:aa:8b (88:c5:f2:e7:aa:8b), Dst: d2:f8:8c:0b:8b:ab (d2:f8:8c:0b:8b:ab)
  Internet Protocol Version 4, Src: 192.168.43.225 (192.168.43.225), Dst: agnigarh.iitg.ac.in (14.139.196.11)
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 52
    Identification: 0xea57 (59991)
    Flags: 0x4000, Don't fragment
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0x914c [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.43.225 (192.168.43.225)
    Destination: agnigarh.iitg.ac.in (14.139.196.11)
  Transmission Control Protocol, Src Port: 46710 (46710), Dst Port: cirrossp (10443), Seq: 46, Ack: 46, Len: 0
```

- **Identification** – If an IP packet is fragmented during the transmission, all the fragments contain the same identification number to identify the original IP packet they belong to.
- **Flags** are required by the network resources, if an IP Packet is too large to handle, these 'flags' tell if they can be fragmented or not. In this 3-bit flag, the MSB is always set to '0'.
- **Fragment Offset** tells the exact position of the fragment in the original IP Packet.
- **Time to Live** - To avoid looping in the network, every packet is sent with some TTL value set, which tells the network how many routers (hops) this packet can cross.
- **Protocol** tells the network layer at the destination host, to which protocol this packet belongs to, i.e. the next level Protocol.
- **Header Checksum** field is used to keep the checksum value of the entire header which is then used to check if the packet is received error-free.
- **Source Address** is a 32-bit address of the sender (or source) of the packet.
- **Destination Address** is a 32-bit address of the Receiver (or destination) of the packet.

4. Link Layer:

a. The Ethernet II packet has the following fields:

- **Destination and Source IP** represent the IP address of endpoints
- **Destination and Source MAC** represent the MAC address and manufacturer details of destination and source
- **Type** tells about the IP version, i.e. IPv4 or IPv6

```

> Frame 60: 189 bytes on wire (1512 bits), 189 bytes captured (1512 bits) on interface wlo1, id 0
> Ethernet II, Src: AzureWav_e7:aa:8b (80:c5:f2:e7:aa:8b), Dst: d2:f8:8c:0b:8b:ab (d2:f8:8c:0b:8b:ab)
  > Destination: d2:f8:8c:0b:8b:ab (d2:f8:8c:0b:8b:ab)
    Address: d2:f8:8c:0b:8b:ab (d2:f8:8c:0b:8b:ab)
    ....1. .... = LG bit: Locally administered address (this is NOT the factory default)
    ....0. .... = IG bit: Individual address (unicast)
  > Source: AzureWav_e7:aa:8b (80:c5:f2:e7:aa:8b)
    Address: AzureWav_e7:aa:8b (80:c5:f2:e7:aa:8b)
    ....0. .... = LG bit: Globally unique address (factory default)
    ....0. .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 192.168.43.225 (192.168.43.225), Dst: agnigarh.iitg.ac.in (14.139.196.11)
> Transmission Control Protocol, Src Port: 46710 (46710), Dst Port: cirsosp (10443), Seq: 276, Ack: 91, Len: 123
> Transport Layer Security

```

Question 2.

FortiClient VPN (Desktop App) has numerous functionalities, some which are freely (and readily) available -

- Establish Connection with Remote Server
- Disconnecting from Remote Server
- Establish SSH connection to a remote PC

However other functionalities like **Endpoint Status (realtime)**, **Endpoint Quarantine (due to security reasons)**, **Software Inventory and Dashboard management** are not available for free. We'll look at the three available features and attempt to explain those.

a. Establish Connection with a Remote Server (using IPv4 over Ethernet II)

- A **DNS** query is sent to **IP addr(destination)** via **UDP**. The destination in our case is **agnigarh.iitg.ac.in**
- On receipt of a response, a **TCP handshake** is done using **ACK** and **SYN** methods
- The **TLS** then shares encrypted keys and source data between endpoints. The **TCP** packets are also exchanged.
- Other protocols involved are **HTTP** (for text files), **OCSP** (for certificates like SHA certificates), **IGMPv3** (for multicasting).

b. Disconnecting from a Remote Server (using IPv4 over Ethernet II)

- **TCP** and **TLS** do their respective jobs before we disconnect, i.e, **TCP** exchanges packets, while **TLS** exchanges security keys
- **NTP** protocol helps in time synchronization and **IGMPv3** helps to disconnect multiple destinations simultaneously.

c. Establishing a SSH connection to a remote PC

- After **TCP** handshake is done between source and remote PCs, sensitive information like passwords and application packets are shared over **TLS** and **TCP** connections respectively.
- **NTP** in this case ensures that the endpoint PCs are synchronised to measure the time delays effectively.
- **HTTP GET** verifies connectivity between the end points.

(For question 3 please find the ..evening files for both types attached)

Question 3.

For this question we'll analyse the 2 functionalities given in the question doc (as sample functionalities), i.e,

- Establish Connection (with agnigarh.iitg.ac.in)

I have attached a screenshot from my laptop (and also added a file **q3_establish_connection_evening.pcapng**), to show my work. As can be seen from the image below, these are the observations:

- In line number 10 (highlighted in blue), the first **DNS** messages are sent to the remote server, to check connectivity.
- Then after a standard query response is received (line 15), the **TCP** starts handshake via **ACK** and **SYN** methods (line 16 and 18 highlighted in gray).
- Then **TLSv1.2** does a '**Client Hello**', and exchanges encrypted and sensitive messages like Client Keys, Cipher Specs etc. (line 29)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	relay-f0fb4e12.net...	...	TCP	66	http(80) → 43279 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=750404...
2	0.000041747	lord	relay-f0fb4e12.net...	TCP	66	[TCP ACKed unseen segment] 43279 → http(80) [ACK] Seq=1 ACK=2...
3	0.453657457	lord	_gateway	DNS	87	Standard query 0x5673 PTR 225.43.168.192.in-addr.arpa
4	0.455945638	_gateway	lord	DNS	105	Standard query response 0x5673 PTR 225.43.168.192.in-addr.arpa...
5	1.489787771	agnigarh.iitg.ac.in	...	TLSv1.2	595	Application Data
6	1.452545938	lord	_gateway	DNS	85	Standard query 0x5c29 PTR 1.43.168.192.in-addr.arpa
7	1.460858595	_gateway	lord	DNS	85	Standard query response 0x5c29 No such name PTR 1.43.168.192...
8	2.451835611	lord	_gateway	DNS	85	Standard query 0x5d75 PTR 11.196.139.14.in-addr.arpa
9	3.987954608	_gateway	lord	DNS	186	Standard query response 0x5d75 PTR 11.196.139.14.in-addr.arpa...
10	3.991352122	lord	_gateway	DNS	79	Standard query response 0x5d75 PTR 11.196.139.14.in-addr.arpa...
11	10.876285219	relay-f0fb4e12.net...	lord	TCP	66	[TCP Dup ACK 1#1] http(80) → 43279 [ACK] Seq=1 Ack=1 Win=501...
12	10.876321565	lord	relay-f0fb4e12.net...	TCP	66	[TCP Dup ACK 2#1] [TCP ACKed unseen segment] 43279 → http(80)...
13	10.876798811	_gateway	lord	DNS	79	Standard query response 0xdf0f AAAA agnigarh.iitg.ac.in
14	10.895232859	lord	_gateway	DNS	79	Standard query response 0xcd41 AAAA agnigarh.iitg.ac.in
15	10.897881156	_gateway	lord	DNS	79	Standard query response 0xcd41 AAAA agnigarh.iitg.ac.in
16	10.898113566	agnigarh.iitg.ac.in	...	TCP	74	56434 → cirtrossp(10443) [SYN] Seq=0 Win=32128 Len=0 MSS=1460...
17	10.960657122	lord	golem.canonical.com	NTP	90	NTP Version 4, client
18	11.062467579	agnigarh.iitg.ac.in	lord	TCP	74	cirtrossp(10443) → 56434 [SYN, ACK] Seq=0 Ack=1 Win=18328 Len=...
19	11.062584329	lord	agnigarh.iitg.ac.in	TCP	66	56434 → cirtrossp(10443) [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSV...
20	11.062657731	lord	agnigarh.iitg.ac.in	TLSv1.2	373	Client Hello
21	11.06272206	lord	agnigarh.iitg.ac.in	TCP	66	56434 → cirtrossp(10443) [ACK] Seq=1 Ack=1 Win=32128 Len=0 MSS=1460...
22	11.619257712	golem.canonical.com	lord	NTP	90	NTP Version 4, server
23	11.619312824	agnigarh.iitg.ac.in	lord	TCP	66	cirtrossp(10443) → 56434 [ACK] Seq=1 Ack=308 Win=19456 Len=0 T...
24	11.619334183	agnigarh.iitg.ac.in	lord	TLSv1.2	764	[TCP Dup ACK 19#1] 56434 → cirtrossp(10443) [ACK] Seq=308 Ack=...
25	11.619358309	lord	agnigarh.iitg.ac.in	TCP	78	[TCP Dup ACK 19#1] 56434 → cirtrossp(10443) [ACK] Seq=308 Ack=...
26	11.619351729	agnigarh.iitg.ac.in	lord	TCP	1514	[TCP Out-Of-Order] cirtrossp(10443) → 56434 [ACK] Seq=1 Ack=30...
27	11.619395641	lord	agnigarh.iitg.ac.in	TCP	66	56434 → cirtrossp(10443) [ACK] Seq=308 Ack=2147 Win=30408 Len=...
28	11.740545516	agnigarh.iitg.ac.in	lord	TCP	78	[TCP Dup ACK 23#1] cirtrossp(10443) → 56434 [ACK] Seq=2147 Ack=...
29	11.758027683	lord	agnigarh.iitg.ac.in	TLSv1.2	192	Client Key Exchange, Change Cipher Spec, Encrypted Handshake ...
30	11.989415294	agnigarh.iitg.ac.in	lord	TCP	66	cirtrossp(10443) → 56434 [ACK] Seq=2147 Ack=434 Win=19456 Len=...
31	11.989458353	agnigarh.iitg.ac.in	lord	TLSv1.2	292	New Session Ticket, Change Cipher Spec, Encrypted Handshake M...
32	11.989511039	lord	agnigarh.iitg.ac.in	TCP	66	56434 → cirtrossp(10443) [ACK] Seq=434 Ack=2373 Win=30408 Len=...
33	11.990102758	lord	agnigarh.iitg.ac.in	TLSv1.2	293	Application Data
34	11.992190850	_gateway	lord	DNS	98	Standard query 0x91c1 A locprod1-elb-eu-west-1.prod.mozaws.net
35	11.992861688	_gateway	lord	DNS	98	Standard query 0x85c3 AAAA locprod1-elb-eu-west-1.prod.mozaws...
36	12.101315342	_gateway	lord	DNS	183	Standard query response 0x85c3 AAAA locprod1-elb-eu-west-1.pr...
37	12.103741361	_gateway	lord	DNS	331	Standard query response 0x91c1 A locprod1-elb-eu-west-1.prod...
38	12.10538329	locprod1-elb-eu-wes...	lord	TCP	74	41048 → https(443) [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK...
39	12.142627118	agnigarh.iitg.ac.in	lord	TLSv1.2	754	[TCP Dup ACK 32#1] 56434 → cirtrossp(10443) [ACK] Seq=661 Ack=...
40	12.142670623	lord	agnigarh.iitg.ac.in	TCP	78	[TCP Dup ACK 32#1] 56434 → cirtrossp(10443) [ACK] Seq=661 Ack=...
41	12.152678467	agnigarh.iitg.ac.in	lord	TCP	1514	[TCP Out-Of-Order] cirtrossp(10443) → 56434 [ACK] Seq=2373 Ack=...
42	12.152722464	lord	agnigarh.iitg.ac.in	TCP	78	56434 → cirtrossp(10443) [ACK] Seq=661 Ack=3821 Win=30408 Len=...
43	12.152845891	agnigarh.iitg.ac.in	lord	TCP	1514	[TCP Out-Of-Order] cirtrossp(10443) → 56434 [ACK] Seq=3821 Ack=...
44	12.153080310	lord	agnigarh.iitg.ac.in	TCP	78	56434 → cirtrossp(10443) [ACK] Seq=661 Ack=5269 Win=30408 Len=...

Frame 10: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on interface wlo1, id 0
 0000 d2 f8 bc 0b 8b 00 c5 f2 e7 aa 8b 08 00 45 00E.
 wireshark_wlo1_20200929192059.oYfYf.pcapng

Packets: 227 / D

No.	Time	Source	Destination	Protocol	Length	Info
19	3.986551877	agnigarh.iitg.ac.in	192.168.43.225	TCP	78	[TCP Dup ACK 14#1] cirtrossp(10443) → 56866 [ACK] Seq=676 Ack=...
20	3.984748528	192.168.43.225	agnigarh.iitg.ac.in	TLSv1.2	174	Application Data
21	4.204866692	agnigarh.iitg.ac.in	192.168.43.225	TCP	66	cirtrossp(10443) → 56866 [ACK] Seq=676 Ack=437 Win=74 Len=0 TS...
22	4.204894405	agnigarh.iitg.ac.in	192.168.43.225	TLSv1.2	209	Application Data
23	4.204912508	192.168.43.225	agnigarh.iitg.ac.in	TCP	66	56866 → cirtrossp(10443) [ACK] Seq=437 Ack=819 Win=30408 Len=0...
24	7.777755676	192.168.43.225	agnigarh.iitg.ac.in	TLSv1.2	111	Application Data
25	8.028403739	agnigarh.iitg.ac.in	192.168.43.225	TLSv1.2	111	Application Data
26	8.028463136	192.168.43.225	agnigarh.iitg.ac.in	TCP	66	56866 → cirtrossp(10443) [ACK] Seq=482 Ack=864 Win=30408 Len=0...
27	8.954583275	192.168.43.225	agnigarh.iitg.ac.in	TLSv1.2	163	Application Data
28	9.978511037	192.168.43.225	agnigarh.iitg.ac.in	TLSv1.2	163	Application Data
29	9.980498852	agnigarh.iitg.ac.in	192.168.43.225	TLSv1.2	163	Application Data
30	9.980523384	192.168.43.225	agnigarh.iitg.ac.in	TCP	66	56866 → cirtrossp(10443) [ACK] Seq=676 Ack=961 Win=30408 Len=0...
31	9.980898081	192.168.43.225	agnigarh.iitg.ac.in	TLSv1.2	155	Application Data
32	9.981525888	192.168.43.225	agnigarh.iitg.ac.in	TLSv1.2	196	Application Data
33	11.130289543	192.168.43.225	agnigarh.iitg.ac.in	TCP	196	[TCP Retransmission] 56866 → cirtrossp(10443) [PSH, ACK] Seq=7...
34	11.264817145	agnigarh.iitg.ac.in	192.168.43.225	TLSv1.2	163	Application Data
35	11.264853614	192.168.43.225	agnigarh.iitg.ac.in	TCP	66	56866 → cirtrossp(10443) [ACK] Seq=895 Ack=1058 Win=30408 Len=...
36	11.264885476	agnigarh.iitg.ac.in	192.168.43.225	TLSv1.2	163	Application Data
37	11.264981778	192.168.43.225	agnigarh.iitg.ac.in	TCP	66	56866 → cirtrossp(10443) [ACK] Seq=895 Ack=1155 Win=30408 Len=...
38	11.264987257	agnigarh.iitg.ac.in	192.168.43.225	TCP	66	cirtrossp(10443) → 56866 [ACK] Seq=1155 Ack=895 Win=79 Len=0 T...
39	11.264917481	agnigarh.iitg.ac.in	192.168.43.225	TLSv1.2	155	Application Data
40	11.264922475	192.168.43.225	agnigarh.iitg.ac.in	TCP	66	56866 → cirtrossp(10443) [ACK] Seq=895 Ack=1244 Win=30408 Len=...
41	11.264926570	agnigarh.iitg.ac.in	192.168.43.225	TLSv1.2	196	Application Data
42	11.264931050	relay-f0fb4e12.net...	192.168.43.225	TCP	66	[TCP Dup ACK 1#1] http(80) → 43279 [ACK] Seq=1 Ack=1 Win=501...
43	11.264938892	192.168.43.225	relay-f0fb4e12.net...	TCP	66	[TCP Dup ACK 2#1] [TCP ACKed unseen segment] 43279 → http(80)...
44	11.265067746	192.168.43.225	agnigarh.iitg.ac.in	TCP	66	56866 → cirtrossp(10443) [ACK] Seq=895 Ack=1374 Win=30408 Len=...
45	11.265462602	192.168.43.225	agnigarh.iitg.ac.in	TLSv1.2	155	Application Data
46	11.265442469	192.168.43.225	agnigarh.iitg.ac.in	TLSv1.2	155	Application Data
47	11.266701762	192.168.43.225	agnigarh.iitg.ac.in	TLSv1.2	1457	Application Data
48	11.266833627	192.168.43.225	agnigarh.iitg.ac.in	TLSv1.2	365	Application Data
49	11.397949324	agnigarh.iitg.ac.in	192.168.43.225	TCP	78	[TCP Dup ACK 38#1] cirtrossp(10443) → 56866 [ACK] Seq=1374 Ack=...
50	11.399502666	agnigarh.iitg.ac.in	192.168.43.225	TCP	66	cirtrossp(10443) → 56866 [ACK] Seq=1374 Ack=984 Win=79 Len=0 T...
51	11.399545008	agnigarh.iitg.ac.in	192.168.43.225	TCP	66	cirtrossp(10443) → 56866 [ACK] Seq=1374 Ack=1073 Win=79 Len=0 ...
52	11.399532075	agnigarh.iitg.ac.in	192.168.43.225	TLSv1.2	1131	Application Data
53	11.399524279	192.168.43.225	agnigarh.iitg.ac.in	TCP	66	56866 → cirtrossp(10443) [ACK] Seq=2763 Ack=2439 Win=30408 Len=...
54	11.400246731	agnigarh.iitg.ac.in	192.168.43.225	TLSv1.2	155	Application Data
55	11.403259796	agnigarh.iitg.ac.in	192.168.43.225	TCP	66	cirtrossp(10443) → 56866 [ACK] Seq=2439 Ack=2763 Win=90 Len=0 ...
56	11.403286896	agnigarh.iitg.ac.in	192.168.43.225	TLSv1.2	155	Application Data
57	11.406637405	192.168.43.225	agnigarh.iitg.ac.in	TLSv1.2	203	Application Data
58	12.038923127	agnigarh.iitg.ac.in	192.168.43.225	TCP	66	cirtrossp(10443) → 56866 [ACK] Seq=2528 Ack=2989 Win=96 Len=0 ...
59	12.038963858	agnigarh.iitg.ac.in	192.168.43.225	TLSv1.2	519	Application Data
60	12.039811241	192.168.43.225	agnigarh.iitg.ac.in	TLSv1.2	155	Application Data
61	12.047013210	192.168.43.225	agnigarh.iitg.ac.in	TLSv1.2	171	Application Data
62	12.198964847	agnigarh.iitg.ac.in	192.168.43.225	TCP	66	cirtrossp(10443) → 56866 [ACK] Seq=2981 Ack=3183 Win=96 Len=0 ...

Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface wlo1, id 0
 0000 80 c5 f2 e7 aa 8b d2 f8 8c 0b 8b ab 08 00 45 00E.
 q3_remote_ssh.pcapng

Packets: 196 / D

and destination.

- d. One peculiar thing in this case is the absence of handshakes which can be attributed to the already complete VPN connection and only SSH remote information sharing.

(For questions 4 and 5 please find the attached 6 trace files and nomenclature is .._morning, .._afternoon and .._evening)

Question 4.

For establish connection functionality

Time of the day(⇒) / Data(↓)	Morning	Afternoon	Evening
a) Throughput	1848 bytes/s	1552 bytes/s	2106 bytes/s

- d. After this TCP application packets are exchanged from line number 30 onwards.

- e. Thereafter DNS and UDP make a connection with the destination (agnigarh.iitg.ac.in) over IPv4 whose IP address is 14.139.196.11.

Yes there are handshaking messages in the sequence, which is obvious because before every TCP application data (packet) sharing, there has to be a handshake done to ensure that there is a smooth data transfer from source to destination and vice versa.

- Establishing a SSH connection to my remote PC (satyapra@172.16.70.72)

I have attached a screenshot in this case as well, showing my work (and as before provided a file q3_remote_ssh_evening.pcapng). These are the observations which can be made by considering the image below.

- a. From the image we see that there is no DNS in this case, but only TCP and TLS. This is mostly because we opened Wireshark after the connection was established over VPN, and then we used SSH to connect to my remote machine.

- b. As is expected TLSv1.2 and TCP are doing their respective jobs in sharing secure data (labelled as application data), and sending packets of information between endpoints.

- c. The two IP addresses in this case that communicate with each other are 192.168.43.225 (IP of my wifi network) and agnigarh.iitg.ac.in, i.e the source

b) RTT (timespan/total packets)	104 ms (avg)	118.7789 ms (avg)	105.5859 ms (avg)
c) Packet Size	192 bytes	184	222 bytes
d) No of packets lost	0	0	0
e) TCP and UDP packets	TCP -> 199 UDP -> 18	TCP -> 171 UDP -> 22	TCP -> 207 UDP -> 18
f) No of responses wrt to 1 request	0.8389	0.8380	0.8907

For ssh remote connection functionality

<u>Time of the day(→) / Data(↓)</u>	<u>Morning</u>	<u>Afternoon</u>	<u>Evening</u>
a) Throughput	1605 bytes/s	1726 bytes/s	1046 bytes/s
b) RTT (timespan/total packets)	107.1984 ms (avg)	98.4317 ms (avg)	165.1224 ms (avg)
c) Packet Size	172 bytes	170 bytes	173 bytes
d) No of packets lost	0	0	0
e) TCP and UDP packets	TCP -> 235 UDP -> 20	TCP -> 221 UDP -> 2	TCP -> 190 UDP -> 0
f) No of responses wrt to 1 request	0.9318	1.0458	0.9191

We found these details as follows :

- **Throughput, RTT, Packet Size (average), Number of packets lost** were found in *Wireshark->Statistics->CaptureFile*
- TCP and UDP packets number was found in *Wireshark->Statistics->Protocol Hierarchy*
- Number of requests wrt to 1 request was found by dividing (after applying filter ***ip.src==192.168.43.225 and ip.dst==192.168.43.225***) the total number of packets coming to (dest) and from (src) host, and taking the ratio. It came close to 1 (which is expected ideally).

Question 5.

2 types of IP addresses were always found and remained constant in both **SSH** and **Establish Connection** traces. They were the host IP and the destination IP addresses corresponding to **192.168.43.225 (host)** and **14.139.196.11(agnigarh.iitg.ac.in)**.

Apart from this several other IP addresses were found particularly the ones related to AWS (amazon web services) and that IP was changing with time. The IP addresses were found in *Wireshark->Statistics->Resolved Addresses->(apply filter hosts)*.

The IPs at different times of day were :

- 52.48.132.232 (compute.amazonaws.com) -> afternoon ssh and establish connection trace
- 54.228.140.0 (prod.mozaws.com) -> evening and morning establish connection trace

This can be explained because the **AWS** server is a cloud server and thus has a dynamic location (not placed at a single geographical location). However **agnigarh** and my **PC** were geographically static and thus had the same IP all day long. Apart from **AWS**, I also found **Canonical (Ubuntu)** servers behaving similarly, because of the same reason.