**Assignment – 2**

**Name: Abhishek Agrahari**

**Roll Number: 190123066**

**Application: GitHub Desktop**

**Traces: https://bit.ly/3BtEJ81**

**Question 1**

The various protocols used by GitHub Desktop application are explained below in the respective layers that they belong to.

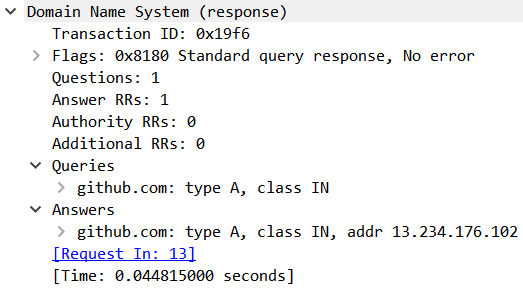
1. **Application Layer** **(Domain Name System):** The Domain Name System is a host name to IP address translation service. It has following fields –

i) Questions: Queries count

ii) Answer RRs: Answers count

iii) Queries: DNS queries for host name resolution (github.com).

iv) Answers: Answer to DNS queries (13.234.176.102).



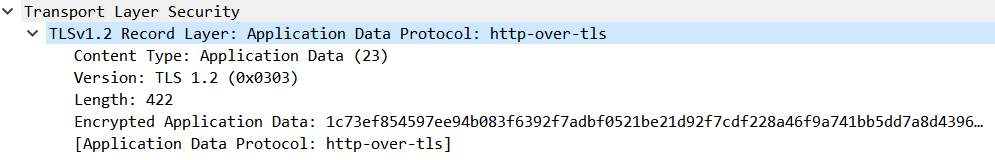
2. **Session Layer**: **Secure Socket Layer, TLSv1.2** –

i) Content Type: Type of content whether Application Data, Handshake etc (Application Data).

ii) Version: TLS 1.2

iii) Length: Length of the data.

iv) Encrypted Application data.



3. **Transport Layer**: **UDP, TCP**

=> TCP (Transmission Control Protocol):

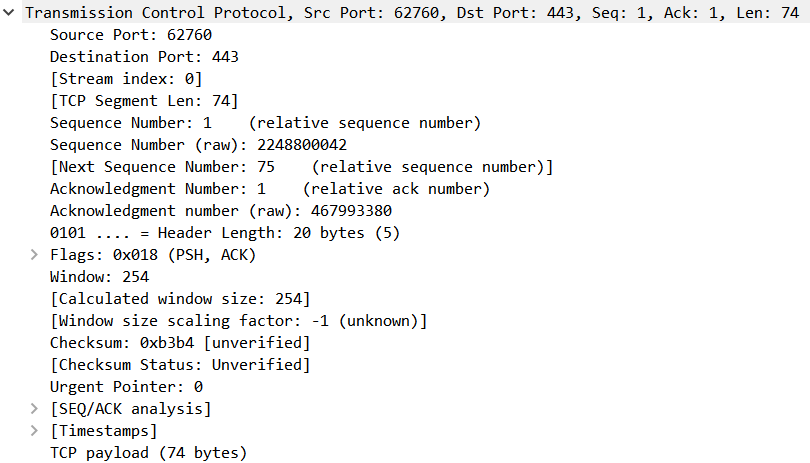
i) Source Port: Port number of the source

ii) Destination Port: Port number of the destination

iii) Sequence Number: byte number of the first byte of data in the TCP packet sent

iv) Acknowledgment number: it is the sequence number of the next byte the receiver expects to receive

v) Checksum: error detection bits of the segment.



=> UDP (User Datagram Protocol):

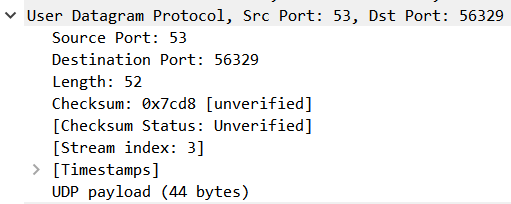
i) Source Port: Port number of the source

ii) Destination Port: Port number of the destination

iii) Length: total length including UDP header and Application

iv) Checksum: It is used to check if data is corrupted or not.

v) Timestamps: Time relative to the last and first frame.



**4. Network Layer: IPv4**

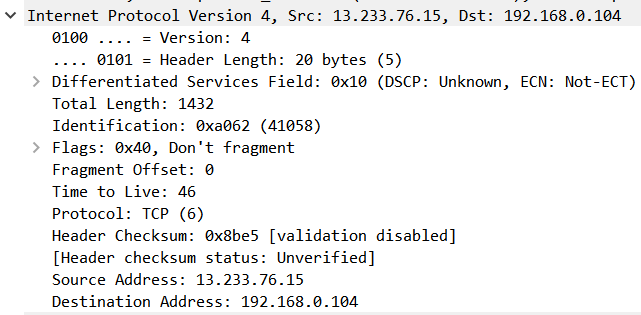
i) Version: Indicates the IP version used

ii) Header Length: contains the length of the IP header

iii) Source: Ip address of the sender.

iv) Destination: Ip address of the receiver.

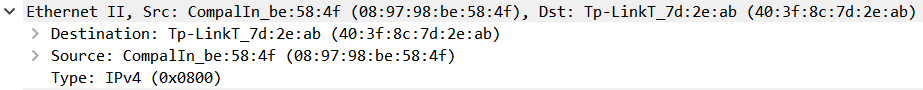
v) Time to Live: maximum number of hops a datagram can take to reach the destination.



**5. Link Layer: Ethernet II**

i) Destination: MAC address of the receiving end.

ii) Source: MAC address of the sending end.

iii) Type: Type of Network Layer Protocol. 

**Question 2**

The important functionalities of GitHub Desktop Application are given below:

(i) Cloning a repository from the internet.

(ii) Adding a local repository to GitHub.

(iii) Pushing a repo onto GitHub server.

(iv) Pulling a repo from GitHub server.

(v) Branching a repository

The **TLS protocol** is used by every functionality of GitHub. TLS protocol is used as it encrypts data to and from the site to clients. This also protects the integrity of the website by helping to prevent intruders tampering between the site and client browsing.

The **DNS protocol** is also used by every functionality of GitHub to resolve the IP address for the github.com. DNS uses UDP packets because these are fast and have low overhead.

The **TCP protocol** is used by all functionalities of the GitHub at the transport layer. TCP always guarantees

that data reaches its destination and it reaches there without duplication. It guarantees reliable data transfer by having handshaking protocol on connection establishment and connection termination.

The **IPv4 protocol** is also used by all functionalities of GitHub at the network layer. It is a connectionless

protocol for use on packet-switched networks. It delivers packets using IP headers from the source to the destination.

The **Ethernet II** is also used by all functionalities of GitHub in the data link layer. This contains information

about source and destination MAC address in its header. Ethernet lying in data link layer is also responsible

for error detection and correction along with ow control. It exists as a point to point connection.

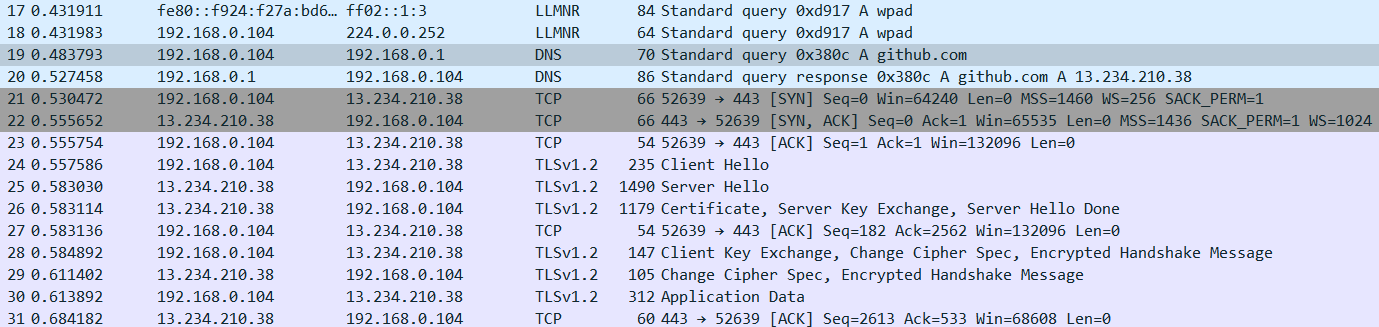
The **User Datagram Protocol** is a connection-less protocol and provides faster data transfer than TCP but

is not very reliable or secure. It is mostly used to transfer small individual packets (like DNS requests). GitHub uses DNS queries to fetch IP addresses, which are made using UDP.

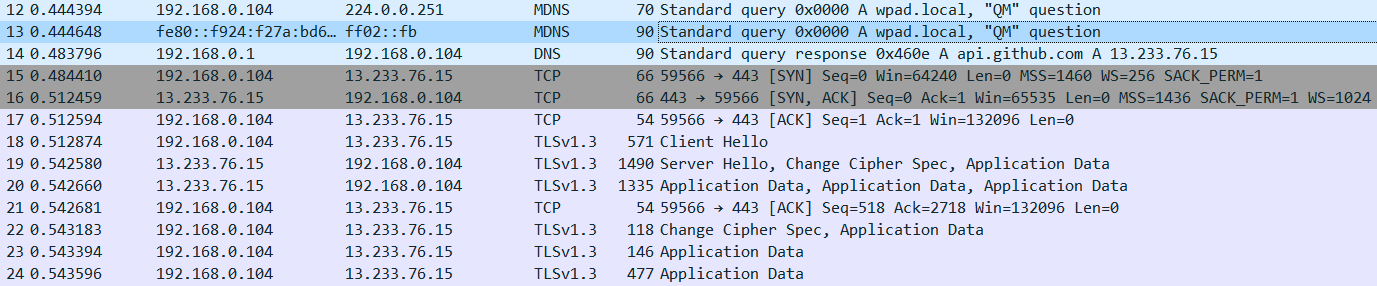
**Question 3**

Two functionalities of GitHub Desktop application along with the sequence of messages exchanged is given below:

**1. Cloning a repository**



**Adding a local repository to GitHub**



Both the above functionalities of GitHub use 3-way handshake for TCP connection establishment and TLS handshaking for further communication.

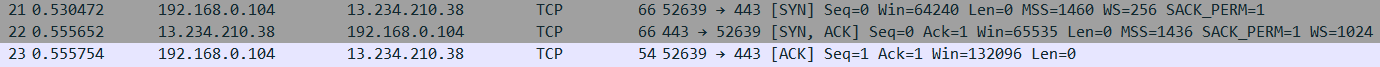
**TCP connection establishment (3-way handshake)**

3-way handshake enables both ends to initiate and negotiate separate TCP socket connections at the same time. It has following three steps:

**Step1(SYN):** Client sends a segment with SYN (Synchronize Sequence Number) which informs the server that the client is likely to start communication and with what sequence number it starts segments with.

**Step2([SYN, ACK]):** Server responds to the client request with SYN-ACK signal bits set. The ACK signifies that the connection request has been acknowledged and SYN signifies the sequence number it is likely to start the segments with.

**Step3(ACK):** The client acknowledges the response of the server and thus establishes a reliable connection with which they start the actual data transfer.



**TLS handshaking**

A TLS handshake starts a communication session that uses TLS encryption. During a TLS handshake, the two communicating sides exchange messages to verify each other, establish the encryption algorithms they will use, and **agree on session keys**. The first message in the TLS Handshake is the **Client Hello** which is sent by the client to initiate a session with the server. In return, the server responds with **Server Hello** and the Server Certificate (for authentication) along with a Server Key, which is used by the client to encrypt Client Key Exchange later in the process**.** Server Hello Done is an indication that the server is now waiting for the client’s response. The client responds with the Client Key and is issued a New Session Ticket. The TLS session is now established, and application data can be exchanged between the server and the client.



**Question 4:**

I have done the cloning operation at three different times of the day.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Time** | **Throughput**  **(in KB/s)** | **RTT (in ms)** | **Avg. Packet Size (Bytes)** | **Packet Lost** | **UDP**  **Packets** | **TCP Packets** | **Number of responses per request** |
| 2:00pm | 80 | 23.62 | 833 | 0 | 42 | 438 | 1.8152 |
| 6:30pm | 109 | 25.28 | 870 | 0 | 17 | 422 | 1.8648 |
| 11:00pm | 59 | 28.20 | 855 | 0 | 21 | 441 | 1.821 |

**Question 5:**

2:00 pm: 13.234.176.102

6:30 pm: 13.234.210.38

11 pm: 13.234.176.102

IP address of GitHub observed at 2pm and 11pm is same but different from one at 6:30 pm. Difference is there because GitHub uses many servers around the world. When a request is sent to GitHub, one of the servers depending on network traffic and congestion a server is assigned for that request. More than one server is used for load balancing and increasing the reliability of the system i.e. even if some server fails traffic can be diverted to other servers.