CS 342 ASSIGNMENT Q4

BY GROUP12 ADITYA GUPTA – 210101009
SWAGAT SATHWARA – 210101102
VATSAL JAIN - 210101110

WIRESHARK REPORT FOR SPOTIFY

INTRODUCTION

This report presents a comprehensive analysis of network traffic related to the Spotify application. The analysis includes protocols used, observe red values for various fields, message sequences, protocol relevance, caching mechanisms, and network statistics.

HOW WE START

First the Wireshark was opened and as we are using the IITG ethernet system, so we open the ethernet in the Wireshark. Then we open the open the app (in our case "Spotify") and do some functions in this app. After this we open command line to find out the IP address of Spotify using the command line **"nslookup spotify.com"** which came out to be **"35.186.224.25".** We set this as our IP address in Wireshark as to study about the packets.

1 TASK 1: PROTOCOLS USED

1.1 APPLICATION LAYER

- **TLS (Transport Layer Security):** TLS is designed to provide security at the transport layer. TLS ensures that no third party may eavesdrop or tamper with any message. A primary use case of TLS is encrypting the communication between web applications and servers, such as web browsers loading a website.

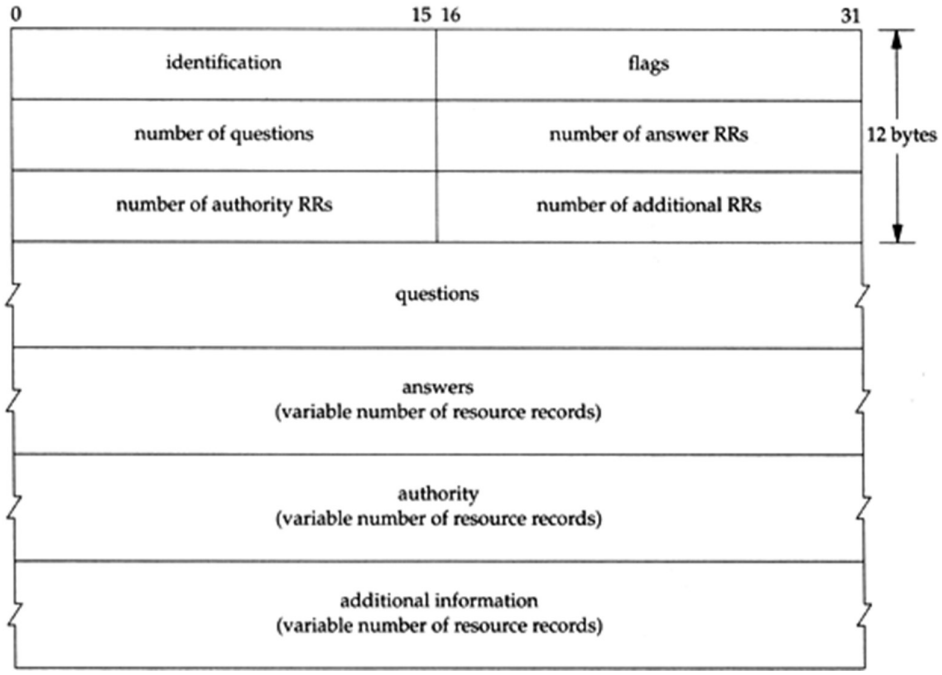
  Packet format of TLS: -

Byte	+0	+1	+2	+3
0	Content type			
1..4	Version		Length	
5..n	*Payload*			
n..m	MAC			
m..p	Padding (block ciphers only)			

  Transport Layer Security (TLS) ensures secure communication on the internet. It uses SSL (Secure Socket Layer) as its basic unit. Each message has a "Content Type" that specifies its purpose (e.g., handshake, data, alert). The "Version" field indicates which TLS version is being used, while "Length" tells us how long the message is. The "Payload" carries the actual data, and the "MAC" ensures data integrity. TLS divides data into records, each with its own MAC for security. Block ciphers, which encrypt fixed-size blocks of data, may require padding

to fill the last block. This ensures data remains confidential and unaltered during transmission, safeguarding online communications effectively.

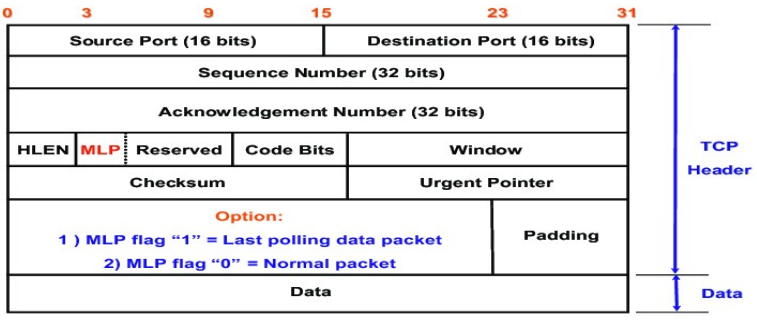
- **DNS (Domain Name System)**: -DNS query to IITG Server was initially used to get IP address of Spotify server, so that packets can filtered basis on that. The client queries an information in a single UDP request. This request is followed by a single UDP reply from the DNS server. A DNS query and a DNS reply share the same structure as shown below: -



Identification field serves as a unique identifier for the DNS query message and is used primarily to match DNS query messages with their corresponding DNS response messages. Flags are used to specify whether it is a query or response. Rest all fields are explained in the diagram above.

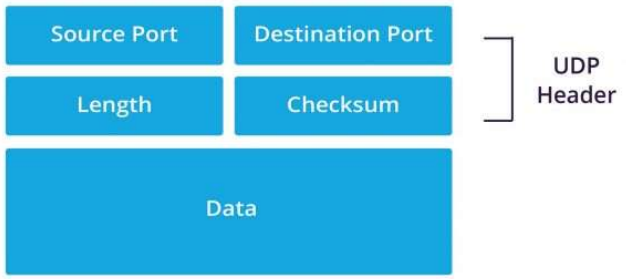
1.2 Transport Layer

- **TCP (Transport Control Protocol)**: - In TCP/IP communication, the Source and Destination Ports pinpoint the originating and receiving applications. The Sequence Number tracks the order of data bytes, while the Acknowledgement Number indicates which data byte the receiver expects next. The Header Length specifies how long the TCP header is. Flags, small indicators like ACK or SYN, denote the type of TCP message. The Checksum ensures data integrity, aiding error correction. Window Size reveals how much data the sender can receive without acknowledgment, crucial for smooth data flow. The Urgent Pointer designates urgent data, counted from the first byte. Options, like timestamps or negotiation parameters, serve various functions, enhancing the protocol's flexibility and capabilities. The packet format of TCP is -



TCP is used in Spotify for Metadata and Control data, Playlist management, User Authentication, File downloads etc.

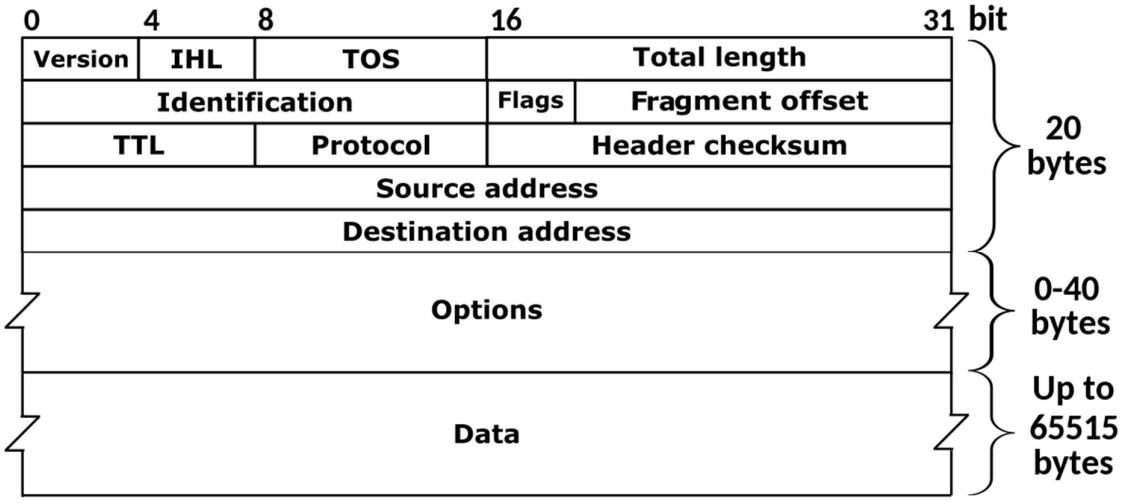
- **UDP (User Datagram Protocol)**: - A UDP (User Datagram Protocol) packet has a simple and efficient structure. It begins with a Source Port field, which identifies the sending application, and a Destination Port field, indicating the receiving application. The Length field specifies the total size of the packet, including the header and data. A checksum is used for error detection, ensuring the integrity of the packet during transmission. Notably, UDP lacks features like sequencing and acknowledgments found in TCP, making it a connectionless protocol, ideal for quick, low-latency data transfers where occasional loss or out-of-order arrival of packets can be tolerated, such as in real-time applications like video streaming or online gaming. UDP packet format is: -



UDP in Spotify is mainly used in Real-Time Streaming, less overhead, Reduce Latency etc.

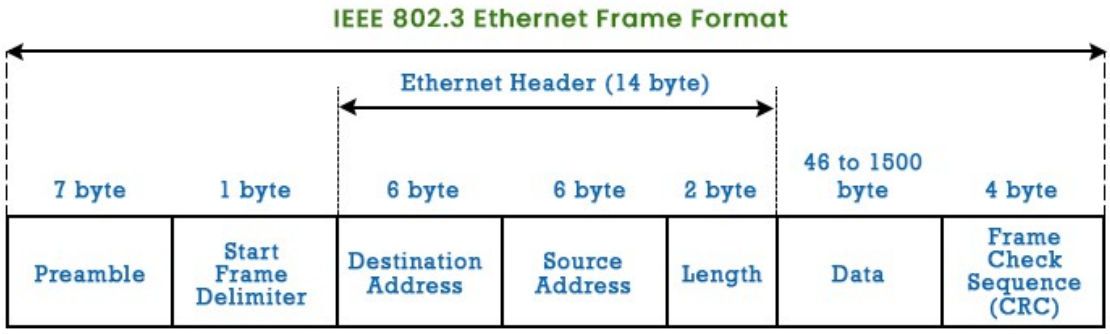
1.3 NETWORK LAYER

- **IPv4(Internet Protocol Version – 4)**: - IPv4 is a widely used protocol over different kinds of network in the Network layer. The structure of IPv4 is shown below: -



1.4 LINK LAYER

- **Ethernet:** - The Ethernet frame starts with a Preamble and SFD (Start Frame Delimiter), which help synchronize devices at the physical layer. The Preamble is a 56-bit pattern of alternating 1s and 0s, aiding in clock alignment. The SFD is an eight-bit signal marking the end of the preamble. Following this, the Destination and Source MAC addresses identify the devices involved. The Ether Type field distinguishes between 802.3 and Ethernet II, indicating the type of Ethernet frame. The Data segment contains the actual information. Finally, the CRC (Cyclic Redundancy Check) employs a CRC-32 code to detect errors in transmission.
  The structure of the ethernet packet:



Proof of the above-mentioned Protocols:

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
∨ Frame	100.0	335	100.0	114839	14 k	0	0	0	335
∨ Ethernet	100.0	335	4.1	4738	616	0	0	0	335
∨ Internet Protocol Version 4	100.0	335	5.8	6700	871	0	0	0	335
› User Datagram Protocol	88.1	295	2.1	2360	307	0	0	0	295
∨ Transmission Control Protocol	11.9	40	13.6	15589	2028	20	1884	245	40
Transport Layer Security	6.0	20	14.2	16277	2118	20	13328	1734	21

2 TASK 2 – OBSERVED VALUES OF VARIOUS FIELDS OF PROTOCOLS

2.1 APPLICATION LAYER

- **TLS (Transport Layer Security): -** In this we observe that the version of TLS used is 1.2, length of packer including header is 185, and content type is application data which is encrypted to ensure authenticity and security.

```
∨ Transport Layer Security
  ∨ TLSv1.2 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 185
    › Handshake Protocol: Client Hello
```

- **DNS (Domain Name System): -** Below given is an example of a DNS query, which uses a UDP protocol in the transport layer for querying the IP address. The query consists of flags which defines the type of DNS query, whether it is a question, recursive or iterative search need to be followed etc. It also defines the number of questions and number of different types of answers (RR's) followed by them in the bottom.

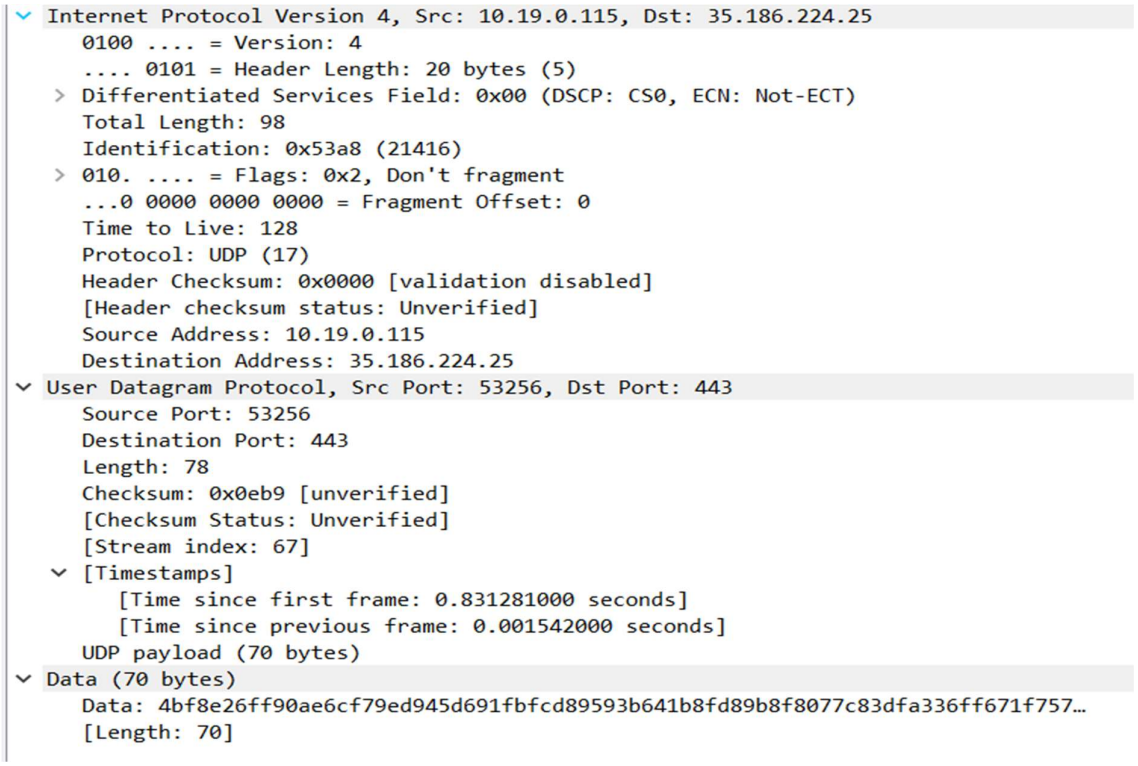
```
User Datagram Protocol, Src Port: 60981, Dst Port: 53
    Source Port: 60981
    Destination Port: 53
    Length: 47
    Checksum: 0xb7d8 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 7]
  > [Timestamps]
    UDP payload (39 bytes)
Domain Name System (query)
    Transaction ID: 0x5e3e
  > 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: 0
  > Queries
      > apresolve.spotify.com: type A, class IN
    [Response In: 227]
```

2.2 TRANSPORT LAYER

- **TCP (Transport Control Protocol): -** We can see that it contains source port, destination port, sequence number, acknowledgment number, header length etc. It also contains flags which contains various metadata like urgent pointer, acknowledgment (whether it is an acknowledgement or not), SYN, FIN, reserved etc. It also contains Window size and checksum (for error detection).

```
Transmission Control Protocol, Src Port: 54235, Dst Port: 443, Seq: 0, Len: 0
    Source Port: 54235
    Destination Port: 443
    [Stream index: 9]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 4135562119
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
  > 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·]
    Window: 64240
    [Calculated window size: 64240]
    Checksum: 0x0e80 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted
  > [Timestamps]
```

- **UDP (User Datagram Protocol)**: -



UDP is used in the process of delivering audio packets from Spotify's servers to your device. We can see that it contains source port, destination port, header length , Time to live , checksum (for error detection).

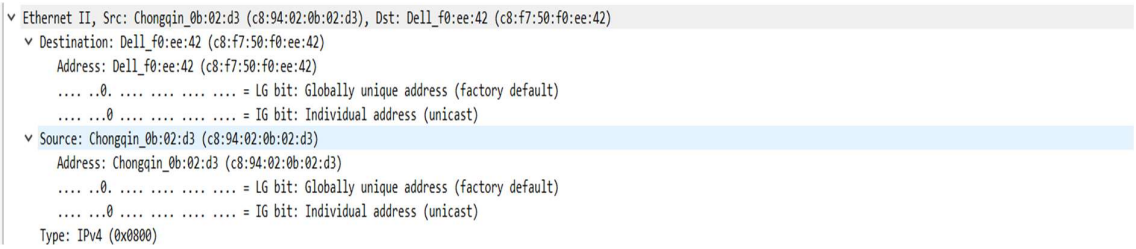
2.3 NETWORK LAYER

- **IPv4: -**

The structure of IPv4 packet is shown above. It has fields containing source IP address, destination IP address, version, header length, total length etc. It also has flags specifying whether it is a reserved bit, fragment offset, whether there are more fragments to come etc (It is used for flow control). It also contains Time to Live (TTL) representing max number of hops of that packet in the network before it is dropped, followed by checksum for error detection at Network level and data payload from above layers.

2.4 LINK LAYER

- **Ethernet -** The structure of ethernet packet is shown above. The Src and Dst field contain the source and destination device's MAC addresses. Type indicates the upper layer protocol used which is IPv4 in this case. The LG bit in both cases is 0, so the addresses are vendor assigned, not administratively assigned. (Here unicast refers to sending data to single destination). This is followed by Data from upper layers.



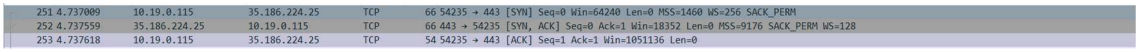
3 TASK 3: TYPES OF MESSAGES EXCHANGED AND HANDSHAKE MECHANISM

NOTE – The QUIC that is mentioned in the below tables is actually a form od UDP known as Quick UDP Internet Connections .

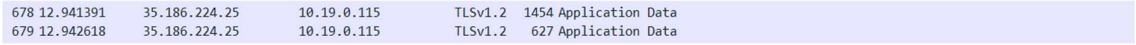
3.1 LAUNCH OF THE APPLICATION

When we launch the application, at first a DNS query is sent to IITG Server to get the IP address of main Spotify server. After that the server responds with the DNS reply and the IP address of the Spotify server is resolved. Followed by this, we observe the conversation, to see that there are 2 hand-shakings that occur.

- **3-way** handshaking in the TCP protocol. Initially, the sender sends an [SYN] (Synchronize Sequence Number) message saying that the client is likely to start communication, and



sends along with-it initial sequence number that the client is planning to use. Server responds by a [SYN, ACK] reply acknowledging the client and telling the client the initial sequence number the server plans to use. Finally, the client acknowledges this also before

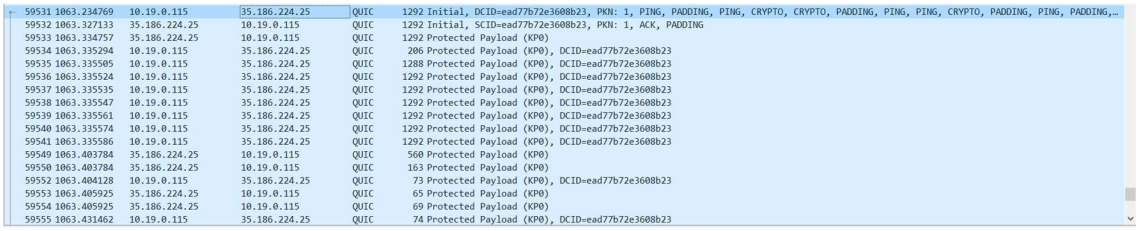


- **2-way** TLS handshake in which the client and server exchange the keys to be used for communication, both the client and host computer agree upon an encryption method from the cipher suites to create keys and encrypt information.

3.2 PLAY A VIDEO

We can observe that packets come in regular intervals of time and the client also sends ACK (acknowledgment) to the server in regular intervals of time. Here we can also notice that the Application data from the server is sent using TLS protocol (adding an extra layer of security)

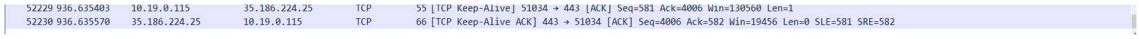
to the client while the ACK (acknowledgment) from the client to server is sent using normal



59531	1063.234769	10.19.0.115	35.186.224.25	QUIC	1292	Initial, DCID=ead77b72e3608b23, PKN: 1, PING, PADDING, PING, CRYPTO, CRYPTO, PADDING, PING, PING, CRYPTO, PADDING, PING, PADDING,…
59532	1063.327133	35.186.224.25	10.19.0.115	QUIC	1292	Initial, SCID=ead77b72e3608b23, PKN: 1, ACK, PADDING
59533	1063.334757	35.186.224.25	10.19.0.115	QUIC	1292	Protected Payload (KP0)
59534	1063.335294	10.19.0.115	35.186.224.25	QUIC	206	Protected Payload (KP0), DCID=ead77b72e3608b23
59535	1063.335505	10.19.0.115	35.186.224.25	QUIC	1288	Protected Payload (KP0), DCID=ead77b72e3608b23
59536	1063.335524	10.19.0.115	35.186.224.25	QUIC	1292	Protected Payload (KP0), DCID=ead77b72e3608b23
59537	1063.335535	10.19.0.115	35.186.224.25	QUIC	1292	Protected Payload (KP0), DCID=ead77b72e3608b23
59538	1063.335547	10.19.0.115	35.186.224.25	QUIC	1292	Protected Payload (KP0), DCID=ead77b72e3608b23
59539	1063.335561	10.19.0.115	35.186.224.25	QUIC	1292	Protected Payload (KP0), DCID=ead77b72e3608b23
59540	1063.335574	10.19.0.115	35.186.224.25	QUIC	1292	Protected Payload (KP0), DCID=ead77b72e3608b23
59541	1063.335586	10.19.0.115	35.186.224.25	QUIC	1292	Protected Payload (KP0), DCID=ead77b72e3608b23
59549	1063.403784	35.186.224.25	10.19.0.115	QUIC	560	Protected Payload (KP0)
59550	1063.403784	35.186.224.25	10.19.0.115	QUIC	163	Protected Payload (KP0)
59552	1063.404128	10.19.0.115	35.186.224.25	QUIC	73	Protected Payload (KP0), DCID=ead77b72e3608b23
59553	1063.405925	35.186.224.25	10.19.0.115	QUIC	65	Protected Payload (KP0)
59554	1063.405925	35.186.224.25	10.19.0.115	QUIC	69	Protected Payload (KP0)
59555	1063.431462	10.19.0.115	35.186.224.25	QUIC	74	Protected Payload (KP0), DCID=ead77b72e3608b23

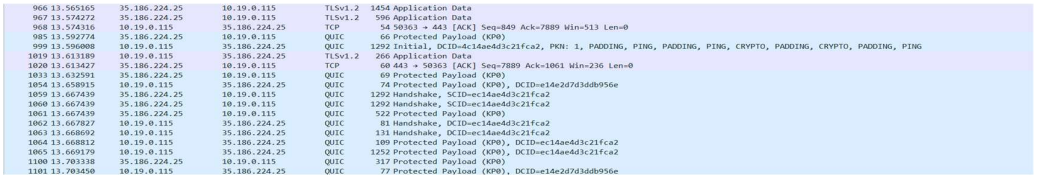
UDP protocol.

3.3 Pause a Video



52229	936.635403	10.19.0.115	35.186.224.25	TCP	55	[TCP Keep-Alive] 51034 → 443 [ACK] Seq=581 Ack=4006 Win=130560 Len=1
52230	936.635570	35.186.224.25	10.19.0.115	TCP	66	[TCP Keep-Alive ACK] 443 → 51034 [ACK] Seq=4006 Ack=582 Win=19456 Len=0 SLE=581 SRE=582

3.4 Skipping The Video

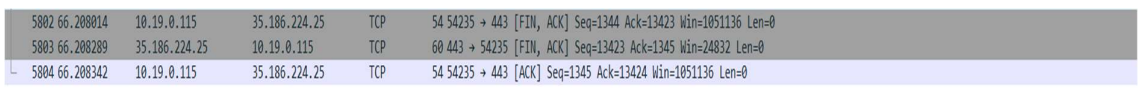


966	13.565165	35.186.224.25	10.19.0.115	TLSv1.2	1454	Application Data
967	13.574272	35.186.224.25	10.19.0.115	TLSv1.2	596	Application Data
968	13.574316	10.19.0.115	35.186.224.25	TCP	54	50363 → 443 [ACK] Seq=849 Ack=7889 Win=513 Len=0
985	13.592774	35.186.224.25	10.19.0.115	QUIC	66	Protected Payload (KP0)
999	13.596008	10.19.0.115	35.186.224.25	QUIC	1292	Initial, DCID=4c14ae4d3c21fca2, PKN: 1, PADDING, PING, PADDING, PING, CRYPTO, PADDING, CRYPTO, PADDING, PING
1019	13.613189	10.19.0.115	35.186.224.25	TLSv1.2	266	Application Data
1020	13.613427	35.186.224.25	10.19.0.115	TCP	60	443 → 50363 [ACK] Seq=7889 Ack=1061 Win=236 Len=0
1033	13.632591	35.186.224.25	10.19.0.115	QUIC	69	Protected Payload (KP0)
1054	13.658915	10.19.0.115	35.186.224.25	QUIC	74	Protected Payload (KP0), DCID=e14e2d7d3ddb956e
1059	13.667439	35.186.224.25	10.19.0.115	QUIC	1292	Handshake, SCID=ec14ae4d3c21fca2
1060	13.667439	35.186.224.25	10.19.0.115	QUIC	1292	Handshake, SCID=ec14ae4d3c21fca2
1061	13.667439	35.186.224.25	10.19.0.115	QUIC	522	Protected Payload (KP0)
1062	13.667827	10.19.0.115	35.186.224.25	QUIC	81	Handshake, DCID=ec14ae4d3c21fca2
1063	13.668692	10.19.0.115	35.186.224.25	QUIC	131	Handshake, DCID=ec14ae4d3c21fca2
1064	13.668812	10.19.0.115	35.186.224.25	QUIC	109	Protected Payload (KP0), DCID=ec14ae4d3c21fca2
1065	13.669179	10.19.0.115	35.186.224.25	QUIC	1252	Protected Payload (KP0), DCID=ec14ae4d3c21fca2
1100	13.703338	35.186.224.25	10.19.0.115	QUIC	317	Protected Payload (KP0)
1101	13.703450	10.19.0.115	35.186.224.25	QUIC	77	Protected Payload (KP0), DCID=e14e2d7d3ddb956e

3.5 Closing The Application

- The above shown example is a case of abrupt exit, where one end exits abruptly and TCP exit handshaking does not occur as a result. There would be retransmissions from the other end for a particular time limit after that it also exits.

- Now coming to graceful exit, we can see a **3-way** handshaking. First client sends to server keeping [FIN, ACK] on indicating that we can finish conversation (FIN bit represents it). After that server acknowledges it by again sending a TCP message keeping [FIN, ACK] on. Then again client just acknowledges it keeping [ACK] on and connection is closed.



5802	66.208014	10.19.0.115	35.186.224.25	TCP	54 54235 → 443 [FIN, ACK] Seq=1344 Ack=13423 Win=1051136 Len=0
5803	66.208289	35.186.224.25	10.19.0.115	TCP	60 443 → 54235 [FIN, ACK] Seq=13423 Ack=1345 Win=24832 Len=0
5804	66.208342	10.19.0.115	35.186.224.25	TCP	54 54235 → 443 [ACK] Seq=1345 Ack=13424 Win=1051136 Len=0

4 TASK 4 – IMPORTANCE OF ALL THE PROTOCOLS USED IN SPOTIFY

4.1 TLS (TRANSPORT LAYER SECURITY)

- **Secure Connection Establishment**: When you open the Spotify app or website, it establishes a secure connection with Spotify's servers using TLS. This ensures that the data exchanged between your device and the servers is encrypted and protected from eavesdropping or tampering.
- **User Authentication**: TLS helps in authenticating the user's device and the Spotify server. This ensures that you are connecting to the legitimate Spotify servers and not to a malicious entity trying to impersonate them.
- **Encryption of Data in Transit**: All data transmitted between your device and Spotify's servers, including your login credentials, search queries, playlists, and song requests, is encrypted using TLS. This safeguards your personal information from unauthorized access.
- **Protection Against Man-in-the-Middle Attacks**: TLS prevents a third-party attacker from intercepting and reading the data exchanged between your device and Spotify's servers. This is crucial in public Wi-Fi networks and other potentially insecure environments.
- **Integrity Checking**: TLS ensures that the data received by your device has not been tampered with during transit. If any modification occurs, it would be detected, and the connection would be terminated.
- **Ensuring Trust in Spotify's Servers**: Spotify's servers present a digital certificate during the TLS handshake process to prove their authenticity. Your device checks this certificate to ensure it's issued by a trusted Certificate Authority (CA). This helps prevent connecting to fake servers.
- **Compliance with Privacy Regulations**: TLS is essential for compliance with data privacy regulations like GDPR, which require secure transmission of personal information.

4.2 DNS (DOMAIN NAME SYSTEM)

- **Resolving Hostnames**: When you open the Spotify app or website, your device needs to know which server to connect to. It does this by sending a DNS query to a DNS server. The query includes the domain name "spotify.com". The DNS server then responds with the corresponding IP address of Spotify's servers.
- **Load Balancing**: Spotify likely employs DNS load balancing to distribute incoming traffic across multiple servers. This ensures that no single server gets overloaded, helping to maintain a smooth user experience.
- **Caching for Performance**: DNS servers often cache resolved domain names for a certain period. This means that if you access Spotify frequently, your DNS server may already have the IP address in its cache, allowing for faster resolution without needing to query again.
- **Redundancy and Failover**: Spotify likely has multiple DNS servers to provide redundancy. If one DNS server is unavailable, your device can try another one to resolve the domain name.
- **CDN Resolution**: Spotify uses Content Delivery Networks (CDNs) to serve content efficiently. DNS is used to resolve the domain names associated with these CDNs, helping to deliver content from servers that are geographically closer to you.
- **DNS Security (DNSSEC**): Spotify may implement DNSSEC to ensure that the DNS responses it receives are authentic and have not been tampered with during transit.

- **Subdomain Resolution**: Spotify may use various subdomains (e.g., play.spotify.com, accounts.spotify.com) for different functionalities. DNS is used to resolve these subdomains to their respective IP addresses.

4.3 TCP (Transport Control Protocol)

- **User Authentication**: When you log in to your Spotify account, TCP is likely used to establish a secure and reliable connection to Spotify's authentication servers. This ensures that your login credentials are transmitted securely.
- **Playlist Management**: When you create, edit, or manage playlists, TCP is used to send your instructions to Spotify's servers. TCP's reliability ensures that your changes are accurately received and processed.
- **Search and Browsing**: When you search for artists, albums, or songs, TCP is used to send your search query to Spotify's servers. TCP ensures that your search results are reliably delivered back to your device.
- **Account Settings**: Any changes you make to your account settings, such as updating your email address or subscription preferences, are likely transmitted using TCP to ensure that the changes are correctly processed by Spotify's servers.
- **Music Downloads**: When you download songs for offline listening, TCP is used to transmit the audio files from Spotify's servers to your device. TCP's reliability is crucial for ensuring that the downloaded files are complete and accurate.
- **Software Updates**: TCP is used to download and install updates for the Spotify application on your device. This ensures that you have the latest features and security patches.
- **Error Correction**: TCP includes mechanisms for error correction, ensuring that if any data packets are lost or corrupted during transmission, they can be retransmitted to guarantee complete and accurate delivery.
- **Maintaining Session State**: TCP helps in maintaining a continuous and stable connection between your device and Spotify's servers. This is crucial for activities that require ongoing interaction, such as streaming playlists or browsing content

4.4 UDP (User Datagram Protocol)

- **Audio Streaming**: When you play a song on Spotify, UDP is employed to transmit the audio data from Spotify's servers to your device. UDP's lower overhead and reduced latency make it ideal for real-time streaming where speed is crucial, and occasional packet loss is acceptable.
- **Low Latency Streaming**: UDP allows for faster transmission of audio packets, reducing the delay between when the audio data is sent and when it's played on your device. This is vital for providing a seamless and responsive listening experience.
- **Handling Real-Time Interactions**: Features like adjusting the volume, skipping tracks, or pausing playback require real-time responsiveness. UDP ensures that these interactions happen promptly without significant delays.
- **Live Broadcasting and Podcasts**: Spotify may use UDP for live broadcasts or podcasts where immediate transmission and synchronization of audio content are paramount.
- **Reduced Overhead**: UDP doesn't have the same level of overhead as TCP, which makes it more efficient for streaming audio where a quick, uninterrupted flow of data is critical.
- **Graceful Handling of Packet Loss**: In real-time streaming, occasional packet loss can occur due to network congestion. UDP's lack of retransmission mechanisms means that lost

packets are not retransmitted. Instead, Spotify likely employs error concealment techniques to mask any minor glitches in audio quality.

- **Voice Communication (Future Features)**: If Spotify were to introduce features like voice communication or live audio chats, UDP would be essential for ensuring real-time, low-latency audio transmission.

4.5 IPv4

- **Addressing**: IPv4 assigns a unique 32-bit numerical address to each device connected to the internet. When you use Spotify, your device (like a smartphone or computer) is assigned an IPv4 address. This address is used to ensure that data packets are sent to the correct destination.
- **Routing:** When you send a request to Spotify's servers, IPv4 helps routers on the internet determine the most efficient path for your data packets to reach their destination. This ensures that your request is delivered to Spotify's servers in an optimal manner.
- **Socket Communication**: IPv4 is used in conjunction with protocols like TCP and UDP to establish connections between your device and Spotify's servers. This enables reliable data transmission for activities like logging in, managing playlists, and streaming music.
- **Network Layer Protocol**: IPv4 operates at the network layer of the OSI model, providing a standardized way for devices to communicate across different networks. This is crucial for enabling global access to Spotify's services.
- **Internet Access:** When you open the Spotify app or website, your device uses IPv4 to connect to your internet service provider (ISP). The ISP then uses IPv4 to forward your requests to Spotify's servers.
- **DNS Resolution**: IPv4 addresses are used in the DNS (Domain Name System) resolution process. When you type "spotify.com" in your web browser or app, DNS translates this human-readable domain name into the corresponding IPv4 address.
- **Compatibility**: IPv4 remains widely used, and the vast majority of internet-connected devices and networks still rely on IPv4. This ensures that Spotify remains accessible to a broad user base.

4.6 Ethernet

- **Local Area Network (LAN) Connectivity**: Within Spotify's offices, data centers, and server farms, Ethernet is the primary means of connecting devices like servers, switches, and storage systems. This allows for high-speed, reliable communication within these local networks.
- **Switching and Routing**: Ethernet switches play a crucial role in directing data packets to their intended destinations within Spotify's internal network. Routers, which operate at the network layer, facilitate communication between different subnetworks within Spotify's infrastructure.
- **Data Centre Connectivity**: In large-scale data centres where Spotify's servers are hosted, Ethernet is used extensively for interconnecting servers, storage devices, and networking equipment. This ensures that data can be efficiently transferred between these components.
- **Load Balancing**: Ethernet-based load balancers distribute incoming network traffic across multiple servers to optimize performance and ensure that no single server gets overloaded. This helps maintain a smooth user experience on the Spotify platform.

- **Redundancy and Failover**: Ethernet's redundancy features ensure that if one network link or component fails, traffic can be automatically rerouted through alternate paths. This helps enhance the reliability of Spotify's services.

5 TASK 5 – CACHE MECHANISM

- **Content Delivery Networks (CDNs):** Spotify likely employs CDNs to cache and deliver content to users. CDNs are distributed servers located around the world that store copies of content (like audio files) closer to the end-users. This reduces the distance data needs to travel, resulting in faster load times.
- **Local Cache on Devices**: Spotify may also utilize local caching on user devices. For instance, when a user listens to a song, it might be temporarily stored on the device's local storage. This allows for quicker playback of frequently accessed songs without the need to fetch them from the server every time.
- **Playlist Caching**: Spotify might cache playlists on the user's device. When a user accesses a playlist they've recently played, it can load faster because the playlist data is already stored locally.
- **Metadata and Album Art:** Information like song titles, artist names, and album art can be cached on the user's device. This improves the responsiveness of the app since it doesn't need to fetch this information from the server repeatedly.
- **User Preferences and Settings**: User settings, preferences, and playlists are likely cached to avoid unnecessary server requests and to provide a smoother user experience.
- **Dynamic Caching for Popular Content**: Popular songs or albums may be dynamically cached in strategic locations to reduce server load and decrease response times for frequently requested content.

6 STATISTICS

File

Name:	C:\Users\Asus\OneDrive\Documents\ad.pcapng
Length:	22 MB
Hash (SHA256):	4bb40e463cba4187fa70b34aa4f56bfae0c914643e9b6e9bee80956bb3349cf6
Hash (RIPEMD160):	b69f6dd2e3d1f771e817bc6d30a27ad34d9c8193
Hash (SHA1):	2b90d1e2c8a1625e1b4bf061749196f57d231fe3
Format:	Wireshark/... - pcapng
Encapsulation:	Ethernet

Time

First packet:	2023-09-10 13:44:28
Last packet:	2023-09-10 15:05:13
Elapsed:	01:20:45

Capture

Hardware:	Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz (with SSE4.2)
OS:	64-bit Windows 10 (22H2), build 19045
Application:	Dumpcap (Wireshark) 4.0.8 (v4.0.8-0-g81696bb74857)

Interfaces

Interface	Dropped packets	Capture filter	Link type	Packet size limit (snaplen)
Ethernet	Unknown	none	Ethernet	262144 bytes

Statistics

Measurement	Captured	Displayed	Marked
Packets	96346	335 (0.3%)	–
Time span, s	4845.363	61.471	–
Average pps	19.9	5.4	–
Average packet size, B	197	343	–
Bytes	18977617	114839 (0.6%)	0
Average bytes/s	3916	1868	–
Average bits/s	31 k	14 k	–

Details

File

Name:	C:\Users\Asus\AppData\Local\Temp\wireshark_EthernetVR90A2.pcapng
Length:	2253 kB
Hash (SHA256):	88ac243e878d76715dc95bacba3a2956d26b74923cb18b559f3622b1dc952817
Hash (RIPEMD160):	d389cda8ef8eaf7876db27a7d0fdc76fc7455e3e
Hash (SHA1):	eecbf83e075801ec4d46cfed802a1a8435e66356
Format:	Wireshark/... - pcapng
Encapsulation:	Ethernet

Time

First packet:	2023-09-10 18:15:27
Last packet:	2023-09-10 18:16:40
Elapsed:	00:01:13

Capture

Hardware:	Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz (with SSE4.2)
OS:	64-bit Windows 10 (22H2), build 19045
Application:	Dumpcap (Wireshark) 4.0.8 (v4.0.8-0-g81696bb74857)

Interfaces

Interface	Dropped packets	Capture filter	Link type	Packet size limit (snaplen)
Ethernet	Unknown	none	Ethernet	262144 bytes

Statistics

Measurement	Captured	Displayed	Marked
Packets	5826	460 (7.9%)	–
Time span, s	73.653	60.739	–
Average pps	79.1	7.6	–
Average packet size, B	354	589	–
Bytes	2059985	270850 (13.1%)	0
Average bytes/s	27 k	4459	–
Average bits/s	223 k	35 k	–