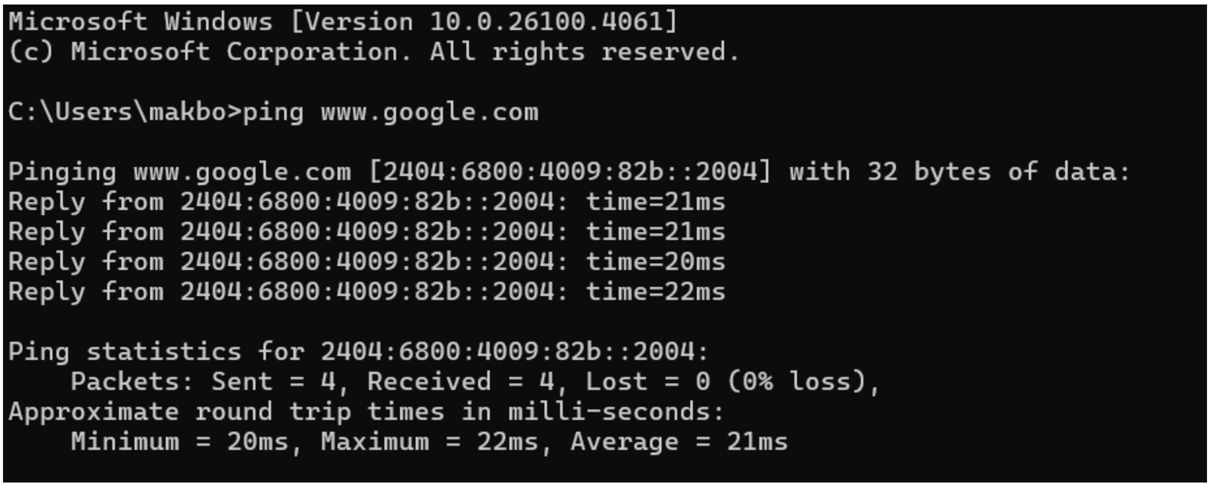
I started Wireshark and opened my command prompt and entered the following command:

ping www.google.com

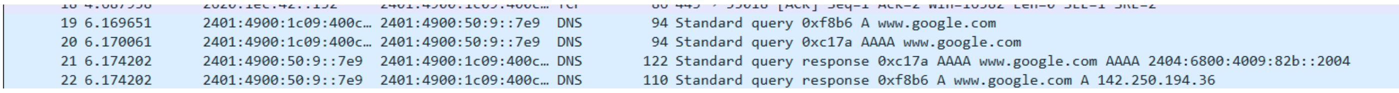


```
Microsoft Windows [Version 10.0.26100.4061]
(c) Microsoft Corporation. All rights reserved.

C:\Users\makbo>ping www.google.com

Pinging www.google.com [2404:6800:4009:82b::2004] with 32 bytes of data:
Reply from 2404:6800:4009:82b::2004: time=21ms
Reply from 2404:6800:4009:82b::2004: time=21ms
Reply from 2404:6800:4009:82b::2004: time=20ms
Reply from 2404:6800:4009:82b::2004: time=22ms

Ping statistics for 2404:6800:4009:82b::2004:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 20ms, Maximum = 22ms, Average = 21ms
```

The ping command is used to test network connectivity between two devices by sending Internet Control Message Protocol (ICMP) Echo Request packets and measuring the response time. It helps diagnose network issues, check reachability, and measure latency.

For sending the echo request packets, it needs the IP address of the website we are trying to reach here which is www.google.com. So, to translate the domain name (in our case google.com) to IP addresses there is another protocol, **DNS which is over UDP port 53**. ***This protocol searches for the domain name on the DNS servers, and returns the IP address.*** The following screenshot from Wireshark displays DNS packets searching (querying) for www.google.com, and response from the servers with the appropriate IP addresses, including both versions (IPv4 and IPv6).

Packets 19 through 22:

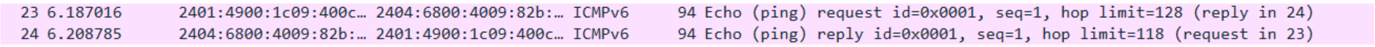


```
19 6.169651    2401:4900:1c09:400c… 2401:4900:50:9::7e9  DNS     94 Standard query 0xf8b6 A www.google.com
20 6.170061    2401:4900:1c09:400c… 2401:4900:50:9::7e9  DNS     94 Standard query 0xc17a AAAA www.google.com
21 6.174202    2401:4900:50:9::7e9  2401:4900:1c09:400c… DNS    122 Standard query response 0xc17a AAAA www.google.com AAAA 2404:6800:4009:82b::2004
22 6.174202    2401:4900:50:9::7e9  2401:4900:1c09:400c… DNS    110 Standard query response 0xf8b6 A www.google.com A 142.250.194.36
```

After finding the IP address of google.com, our ping command now sends ICMP packets to it.

ICMP (Internet Control Message Protocol): This is the main protocol used by ping. It handles error messages and operational queries in a network. Ping sends ICMP Echo Request packets, and the target device responds with ICMP Echo Reply packets.

Packets 23 and 24 are the ICMP packets:



```
23 6.187016    2401:4900:1c09:400c… 2404:6800:4009:82b:… ICMPv6    94 Echo (ping) request id=0x0001, seq=1, hop limit=128 (reply in 24)
24 6.208785    2404:6800:4009:82b:… 2401:4900:1c09:400c… ICMPv6    94 Echo (ping) reply id=0x0001, seq=1, hop limit=118 (request in 23)
```

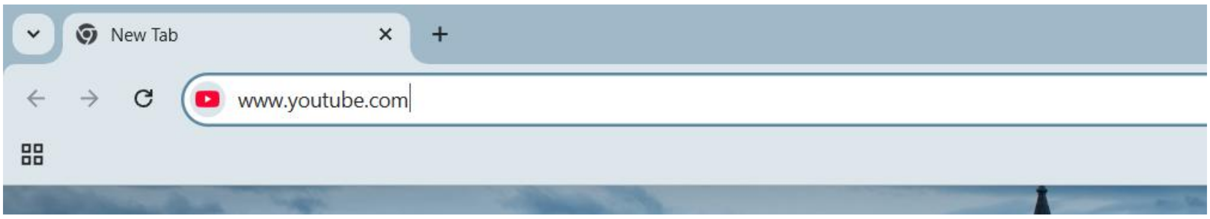
This confirms that the target device (google.com) is **reachable** and **responsive** on the network. It indicates that:

- The destination IP address is active and not blocked by a firewall.
- The network route between sender and receiver is functioning.

- The device at the target address can process ICMP requests.

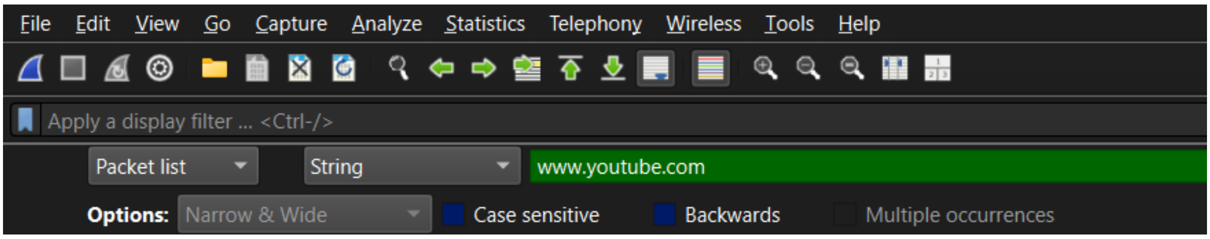
Then, I opened my chrome browser and searched for www.youtube.com. The Wireshark was still on.

The following screenshot documents this:



www.youtube.com

As Wireshark is capturing a lot of live network traffic, to filter out for the step executed above, I searched captured packets for the string "www.youtube.com", under the edit tab, opened the find packets option:

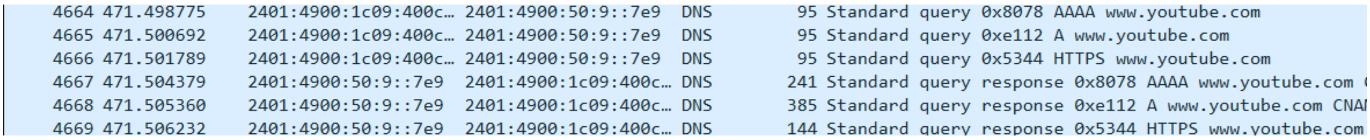


Packet list | String | www.youtube.com

Options: Narrow & Wide □ Case sensitive □ Backwards □ Multiple occurrences

This returned all the packets containing the string – 'www.youtube.com'. As explained in the document above, the domain name is translated to IP address, using the protocol DNS. It can be seen here, how our search query in the browser is handled:

Packets 4664 through 4669 are DNS query requests and responses for youtube.com:



```
4664 471.498775    2401:4900:1c09:400c… 2401:4900:50:9::7e9   DNS        95 Standard query 0x8078 AAAA www.youtube.com
4665 471.500692    2401:4900:1c09:400c… 2401:4900:50:9::7e9   DNS        95 Standard query 0xe112 A www.youtube.com
4666 471.501789    2401:4900:1c09:400c… 2401:4900:50:9::7e9   DNS        95 Standard query 0x5344 HTTPS www.youtube.com
4667 471.504379    2401:4900:50:9::7e9  2401:4900:1c09:400c… DNS       241 Standard query response 0x8078 AAAA www.youtube.com
4668 471.505360    2401:4900:50:9::7e9  2401:4900:1c09:400c… DNS       385 Standard query response 0xe112 A www.youtube.com CNA
4669 471.506232    2401:4900:50:9::7e9  2401:4900:1c09:400c… DNS       144 Standard query response 0x5344 HTTPS www.youtube.com
```

Now, that the IP addresses are known, it is time for a TCP connection.

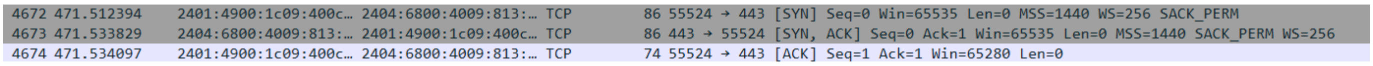
TCP is a connection-oriented protocol used for reliable communication over a network. It ensures data integrity, error recovery, and ordered delivery of packets between sender and receiver. The connection is initiated through a handshake a three-way process.

To establish a connection, TCP uses a **three-step process** between the client and server:

1. **SYN (Synchronize)**: The client sends a SYN packet to initiate a connection. ***Packet: 4672***
2. **SYN-ACK (Synchronize-Acknowledge)**: The server responds with a SYN-ACK packet, acknowledging the request. ***Packet: 4673***

3. **ACK (Acknowledge)**: The client sends an ACK packet, confirming the handshake. ***Packet: 4674***

Once complete, a **reliable, full-duplex communication channel** is established. This ensures **data can flow without loss or corruption**. The packets mentioned in the handshake process above are referenced from here in the following screenshot:



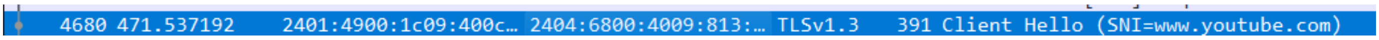
```
4672 471.512394    2401:4900:1c09:400c… 2404:6800:4009:813:… TCP         86 55524 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM
4673 471.533829    2404:6800:4009:813:… 2401:4900:1c09:400c… TCP         86 443 → 55524 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 SACK_PERM WS=256
4674 471.534097    2401:4900:1c09:400c… 2404:6800:4009:813:… TCP         74 55524 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0
```

Now, that a TCP connection is open. The devices are ready to share information. To execute this in a secure way, another protocol is used. TLS – Transport Layer Security.

TLS is a cryptographic protocol that provides **secure communication** over a network by encrypting data between a client (our device) and a server (Youtube's server). It protects against eavesdropping, tampering, and forgery.

The following is TLSv1.3 Client Hello packet which marks the initial step of a TLS 1.3 handshake.

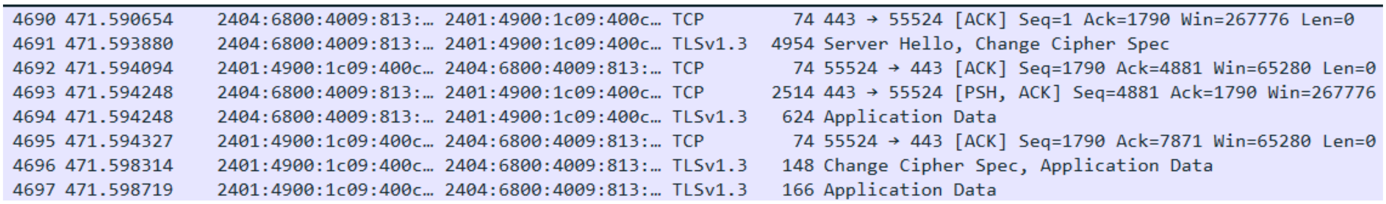
Packet 4680 (TLS Handshake):



```
4680 471.537192    2401:4900:1c09:400c… 2404:6800:4009:813:… TLSv1.3   391 Client Hello (SNI=www.youtube.com)
```

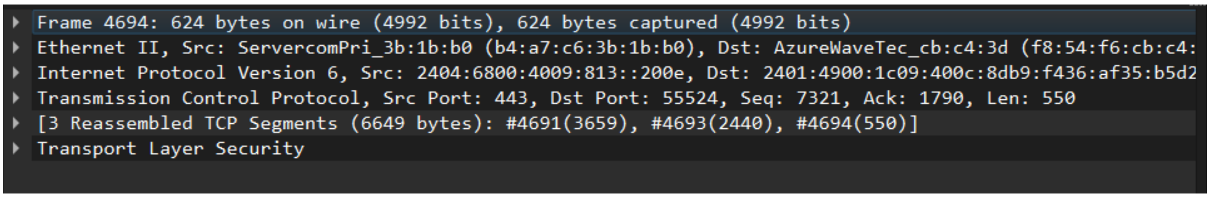
TLS is widely used for securing web traffic (**HTTPS**), emails, and VPNs. In our case it is securing HTTPS web traffic. Now, how de we know whether it is HTTPS traffic or not?

To verify, I clicked on a packet from the below mentioned packets.



```
4690 471.590654    2404:6800:4009:813:… 2401:4900:1c09:400c… TCP        74 443 → 55524 [ACK] Seq=1 Ack=1790 Win=267776 Len=0
4691 471.593880    2404:6800:4009:813:… 2401:4900:1c09:400c… TLSv1.3  4954 Server Hello, Change Cipher Spec
4692 471.594094    2401:4900:1c09:400c… 2404:6800:4009:813:… TCP        74 55524 → 443 [ACK] Seq=1790 Ack=4881 Win=65280 Len=0
4693 471.594248    2404:6800:4009:813:… 2401:4900:1c09:400c… TCP      2514 443 → 55524 [PSH, ACK] Seq=4881 Ack=1790 Win=267776
4694 471.594248    2404:6800:4009:813:… 2401:4900:1c09:400c… TLSv1.3   624 Application Data
4695 471.594327    2401:4900:1c09:400c… 2404:6800:4009:813:… TCP        74 55524 → 443 [ACK] Seq=1790 Ack=7871 Win=65280 Len=0
4696 471.598314    2401:4900:1c09:400c… 2404:6800:4009:813:… TLSv1.3   148 Change Cipher Spec, Application Data
4697 471.598719    2401:4900:1c09:400c… 2404:6800:4009:813:… TLSv1.3   166 Application Data
```

After clicking on one of the packets, Wireshark shows more information on the selected packet as follows:



```
▸ Frame 4694: 624 bytes on wire (4992 bits), 624 bytes captured (4992 bits)
▸ Ethernet II, Src: ServercomPri_3b:1b:b0 (b4:a7:c6:3b:1b:b0), Dst: AzureWaveTec_cb:c4:3d (f8:54:f6:cb:c4:
▸ Internet Protocol Version 6, Src: 2404:6800:4009:813::200e, Dst: 2401:4900:1c09:400c:8db9:f436:af35:b5d2
▸ Transmission Control Protocol, Src Port: 443, Dst Port: 55524, Seq: 7321, Ack: 1790, Len: 550
▸ [3 Reassembled TCP Segments (6649 bytes): #4691(3659), #4693(2440), #4694(550)]
▸ Transport Layer Security
```

By further inspecting it is deduced that this packet has the Source Port: 443. The service that runs at TCP port 443 is Hyper Text Transfer Protocol Secure HTTPS. This is used to transfer Web data over the internet (The Global Network). The packets 4690 through 4697, are all HTTPS traffic. If I click on an any of these packets, it will be found that either the source or the destination port is 443.

More on **HTTPS (HyperText Transfer Protocol Secure)**

HTTPS is a **secure version** of HTTP that encrypts data between a web browser and a server using **TLS (Transport Layer Security)**. It protects against eavesdropping, data tampering, and man-in-the-middle attacks.

How HTTPS Works:

- **Encryption**: TLS encrypts data to ensure privacy.
- **Authentication**: Websites use SSL/TLS certificates to prove their identity.
- **Data Integrity**: Prevents modification of data during transmission.

HTTPS is widely used for **secure web browsing**, login forms, banking transactions, and e-commerce.

So, when I searched YouTube on my browser, and opened a video the packets 4690 through 4697 and beyond those are mostly packets containing the data needed to play that particular video on my device securely, until the connection is closed.