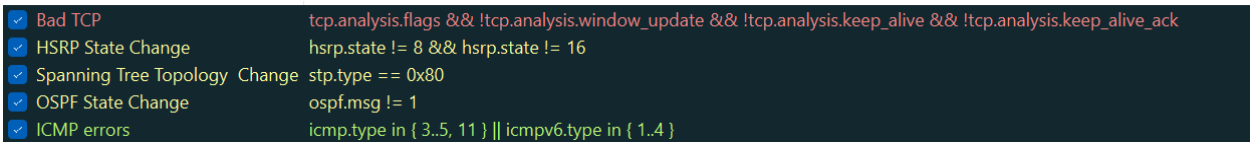
PART - 1

1) Black colour can mean any one of the following(5):



This information about Coloring rules can be seen by going to View => Coloring Rules. Black colour identifies TCP packets with problems and issues like packets being delivered out-of-order.

2) To filter the list of outgoing http traffic:

-> http.request.method == "GET" or http.request.method == "POST"

or we can even simply put filter: http

3) Why does DNS use Follow UDP stream?: DNS (Domain Name System) uses UDP (User Datagram Protocol) because UDP is a connectionless protocol that does not establish a persistent connection between the sender and receiver and DNS queries and responses are typically short-lived and can fit within a single UDP packet. Using Follow UDP Stream allows us to follow the flow of communication for a specific DNS transaction, thus making it faster and more efficient.

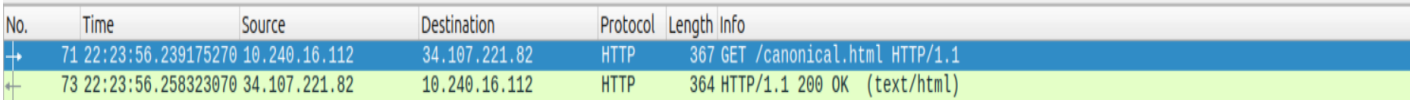
Why does HTTP use Follow TCP stream?: HTTP (Hypertext Transfer Protocol) uses TCP (Transmission Control Protocol) as TCP is a connection-oriented protocol that establishes a reliable, ordered, and error-checked connection between the sender and receiver and HTTP transactions involve multiple packets exchanged between the client and server. Using Follow TCP Stream in Wireshark lets you see the entire conversation between the client and server for a specific HTTP connection.

PART – 2

Using www.amazon.in

1) TCP = Transmission Control Protocol,
HTTP = Hyper Text Transfer Protocol,
OCSP = Online Certificate Status Protocol,
TLSv1.2, TLSv1.3= Transport Layer Security,
DNS = Domain Name Server,
ICMPv6 = Internet Control Message Protocol,
QUIC = Quick UDP Internet Connections,
SSLv2 = Secure Sockets Layer,
ARP = Address Resolution Protocol.

2) Time taken from HTTP GET message to HTTP OK reply = 22:23:56.258323070 - 22:23:56.239175270 = 0.19147800 seconds



No.	Time	Source	Destination	Protocol	Length	Info
71	22:23:56.239175270	10.240.16.112	34.107.221.82	HTTP	367	GET /canonical.html HTTP/1.1
73	22:23:56.258323070	34.107.221.82	10.240.16.112	HTTP	364	HTTP/1.1 200 OK (text/html)

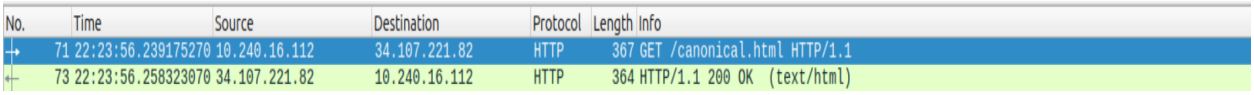
3) Internet address of my computer= 10.240.16.112
IP address of URL = 34.107.221.82

4) No 71,83 is HTTP GET message and No 73,88 is HTTP OK message

```
No.     Time               Source                Destination           Protocol Length Info
     71 22:23:56.239175270 10.240.16.112         34.107.221.82         HTTP     367    GET /canonical.html
HTTP/1.1
Frame 71: 367 bytes on wire (2936 bits), 367 bytes captured (2936 bits) on interface wlp0s20f3, id 0
Ethernet II, Src: IntelCor_1f:79:8a (20:c1:9b:1f:79:8a), Dst: Cisco_0a:9a:e1 (44:b6:be:0a:9a:e1)
Internet Protocol Version 4, Src: 10.240.16.112, Dst: 34.107.221.82
Transmission Control Protocol, Src Port: 43596, Dst Port: 80, Seq: 1, Ack: 1, Len: 301
Hypertext Transfer Protocol
No.     Time               Source                Destination           Protocol Length Info
     73 22:23:56.258323070 34.107.221.82         10.240.16.112         HTTP     364    HTTP/1.1 200
OK  (text/html)
Frame 73: 364 bytes on wire (2912 bits), 364 bytes captured (2912 bits) on interface wlp0s20f3, id 0
Ethernet II, Src: Cisco_13:2a:c2 (f8:7a:41:13:2a:c2), Dst: IntelCor_1f:79:8a (20:c1:9b:1f:79:8a)
Internet Protocol Version 4, Src: 34.107.221.82, Dst: 10.240.16.112
Transmission Control Protocol, Src Port: 80, Dst Port: 43596, Seq: 1, Ack: 302, Len: 298
Hypertext Transfer Protocol
Line-based text data: text/html (1 lines)
No.     Time               Source                Destination           Protocol Length Info
     83 22:23:56.305408151 10.240.16.112         34.107.221.82         HTTP     369    GET /success.txt?
ipv4 HTTP/1.1
Frame 83: 369 bytes on wire (2952 bits), 369 bytes captured (2952 bits) on interface wlp0s20f3, id 0
Ethernet II, Src: IntelCor_1f:79:8a (20:c1:9b:1f:79:8a), Dst: Cisco_0a:9a:e1 (44:b6:be:0a:9a:e1)
Internet Protocol Version 4, Src: 10.240.16.112, Dst: 34.107.221.82
Transmission Control Protocol, Src Port: 43608, Dst Port: 80, Seq: 1, Ack: 1, Len: 303
Hypertext Transfer Protocol
No.     Time               Source                Destination           Protocol Length Info
     88 22:23:56.323078790 34.107.221.82         10.240.16.112         HTTP     282    HTTP/1.1 200
OK  (text/plain)
Frame 88: 282 bytes on wire (2256 bits), 282 bytes captured (2256 bits) on interface wlp0s20f3, id 0
Ethernet II, Src: Cisco_13:2a:c2 (f8:7a:41:13:2a:c2), Dst: IntelCor_1f:79:8a (20:c1:9b:1f:79:8a)
Internet Protocol Version 4, Src: 34.107.221.82, Dst: 10.240.16.112
Transmission Control Protocol, Src Port: 80, Dst Port: 43608, Seq: 1, Ack: 304, Len: 216
Hypertext Transfer Protocol
Line-based text data: text/plain (1 lines)
```

5)

These results from above were obtained from Firefox.



No.	Time	Source	Destination	Protocol	Length	Info
71	22:23:56.239175270	10.240.16.112	34.107.221.82	HTTP	367	GET /canonical.html HTTP/1.1
73	22:23:56.258323070	34.107.221.82	10.240.16.112	HTTP	364	HTTP/1.1 200 OK (text/html)

```
No.     Time               Source                Destination           Protocol Length Info
     71 22:23:56.239175270 10.240.16.112         34.107.221.82         HTTP     367    GET /canonical.html
HTTP/1.1
Frame 71: 367 bytes on wire (2936 bits), 367 bytes captured (2936 bits) on interface wlp0s20f3, id 0
Ethernet II, Src: IntelCor_1f:79:8a (20:c1:9b:1f:79:8a), Dst: Cisco_0a:9a:e1 (44:b6:be:0a:9a:e1)
Internet Protocol Version 4, Src: 10.240.16.112, Dst: 34.107.221.82
Transmission Control Protocol, Src Port: 43596, Dst Port: 80, Seq: 1, Ack: 1, Len: 301
Hypertext Transfer Protocol
No.     Time               Source                Destination           Protocol Length Info
     73 22:23:56.258323070 34.107.221.82         10.240.16.112         HTTP     364    HTTP/1.1 200
OK  (text/html)
Frame 73: 364 bytes on wire (2912 bits), 364 bytes captured (2912 bits) on interface wlp0s20f3, id 0
Ethernet II, Src: Cisco_13:2a:c2 (f8:7a:41:13:2a:c2), Dst: IntelCor_1f:79:8a (20:c1:9b:1f:79:8a)
Internet Protocol Version 4, Src: 34.107.221.82, Dst: 10.240.16.112
Transmission Control Protocol, Src Port: 80, Dst Port: 43596, Seq: 1, Ack: 302, Len: 298
Hypertext Transfer Protocol
Line-based text data: text/html (1 lines)
No.     Time               Source                Destination           Protocol Length Info
     83 22:23:56.305408151 10.240.16.112         34.107.221.82         HTTP     369    GET /success.txt?
ipv4 HTTP/1.1
Frame 83: 369 bytes on wire (2952 bits), 369 bytes captured (2952 bits) on interface wlp0s20f3, id 0
Ethernet II, Src: IntelCor_1f:79:8a (20:c1:9b:1f:79:8a), Dst: Cisco_0a:9a:e1 (44:b6:be:0a:9a:e1)
Internet Protocol Version 4, Src: 10.240.16.112, Dst: 34.107.221.82
Transmission Control Protocol, Src Port: 43608, Dst Port: 80, Seq: 1, Ack: 1, Len: 303
Hypertext Transfer Protocol
No.     Time               Source                Destination           Protocol Length Info
     88 22:23:56.323078790 34.107.221.82         10.240.16.112         HTTP     282    HTTP/1.1 200
OK  (text/plain)
Frame 88: 282 bytes on wire (2256 bits), 282 bytes captured (2256 bits) on interface wlp0s20f3, id 0
Ethernet II, Src: Cisco_13:2a:c2 (f8:7a:41:13:2a:c2), Dst: IntelCor_1f:79:8a (20:c1:9b:1f:79:8a)
Internet Protocol Version 4, Src: 34.107.221.82, Dst: 10.240.16.112
Transmission Control Protocol, Src Port: 80, Dst Port: 43608, Seq: 1, Ack: 304, Len: 216
Hypertext Transfer Protocol
Line-based text data: text/plain (1 lines)
```

But, when tried with both Google Chrome and Brave browsers, no HTTP packet was captured by wireshark

We see this blank list for both chrome and brave browser. The following may be some of the reasons why we are not able to see http protocol packets:

1. Encrypted Traffic (HTTPS): Chrome and Brave use HTTPS (SSL/TLS) to encrypt the data transmitted between the browser and the server. Wireshark won't be able to decrypt this traffic unless you have the private key for the SSL/TLS connection, which is typically not accessible. You might see only encrypted data in the packets.
2. Browser Configuration: Some browsers have security features that can prevent capturing of certain types of traffic. For example, Chrome has features like QUIC (Quick UDP Internet Connections) that may not be fully supported or captured by Wireshark.
3. Browser Preferences: These browsers might have preferences or settings related to network protocols or proxy configurations that could impact packet capture. Check the browser settings to ensure it allows the capturing of network packets.
4. Protocol or Port Changes: These browsers are constantly evolving, and new protocols or changes in the default ports they use might impact the visibility of certain packets. Check the documentation of the browsers for any recent changes.