INT 301

OPEN-SOURCE TECHNOLOGIES

CA-3

SECTION: KE008

Student Details:

Roll no.	Name	Registration no.
17	Avula Pavan Kumar	11906950

Submitted To: Rajeshwar Sharma

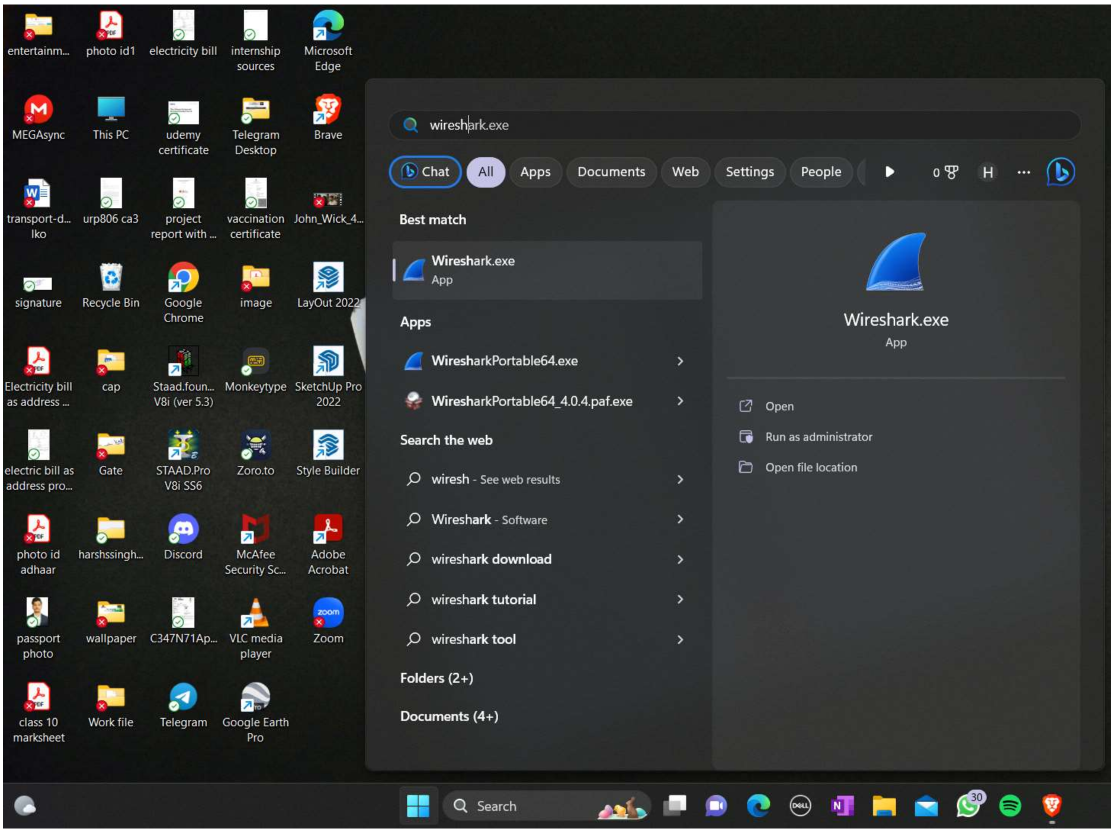
Lovely Professional University

(Jalandhar, Punjab, India)

Table Of Content

28. Implement a network miner tool to detect the operating system, sessions and open ports through packet sniffing and investigate the network traffic.

Introduction



The project at hand aims to implement a network miner tool using Wireshark, an open-source packet analyzer, to detect the operating system, sessions, and open ports through packet sniffing and investigate network traffic. Wireshark allows the user to capture and analyze network traffic in real-time, providing valuable insights into the devices, protocols, and services being used. The project entails using Wireshark to monitor and analyze the packets flowing through the network, which helps to identify the operating system being used by each device on the network. This will involve analyzing the various network protocols and fingerprinting techniques to determine the specific operating system version and vendor. The project also involves capturing and analyzing various network sessions taking place on the network to detect sessions and analyzing the various network packets to identify the ports that are being used by various devices on the network to detect open ports. By implementing this network miner tool, the user can detect the operating system, sessions, and open ports through packet sniffing and investigate network traffic. The user can use Wireshark to determine the operating system being used on a network by analyzing the TTL values of the captured packets. The TTL field in the IP header specifies the maximum number of hops a packet can take before being discarded. Each operating system sets a default TTL value for outgoing packets, which can be used to identify the operating system. The user can also use the TTL field in the IP header to identify TCP sessions in Wireshark by following specific steps. The project also includes finding open ports in Wireshark using the "Statistics" menu by selecting the "Conversations" option and then the "TCP" tab. The user can then sort the list based on the services used during the conversations and look for the services that are using a high number of ports, which are the services that are likely to have multiple open ports. The user can select the service they want to investigate, click on the "Follow" button to view the details of the selected conversation and note down the ports that were used for the selected service. The above steps help the user to identify the open ports in Wireshark. Overall, the project aims to provide a robust network miner tool to detect the operating system, sessions, and open ports through packet sniffing and investigate network traffic using Wireshark.

The objective of the project

- To detect the operating system being used by each device on the network through packet sniffing and analyzing network traffic using Wireshark.
- To detect the network sessions taking place on the network by analyzing network protocols such as TCP, UDP, and ICMP and identifying the start and end of each session, the data being transmitted, and the devices involved in the session.
- To detect the open ports being used by various devices on the network by analyzing the network packets and identifying the ports being used by different services and applications.

The scope of the project are:

- Detect the operating system: One of the objectives of this project is to detect the operating system being used by each device on the network. This is achieved by analyzing the packets using Wireshark and identifying the specific operating system version and vendor by analyzing the various network protocols and fingerprinting techniques.
- Detect sessions: Another objective of the project is to detect the various network sessions taking place on the network. This involves analyzing the various network protocols such as TCP, UDP, and ICMP to identify the start and end of each session, the data being transmitted, and the devices involved in the session.
- Detect open ports: The third objective of the project is to detect the ports that are being used by various devices on the network. This is done by analyzing the various network packets to identify the ports that are being used by different services and applications using

Wireshark.

Dependencies

Access to network traffic data in PCAP format.

Wireshark (or other packet sniffing software) for capturing network traffic.

A stable and fast internet connection for downloading required packages and libraries.

Sufficient storage space for storing captured network traffic and analysis results.

A computer with enough processing power to handle large network traffic datasets and perform analyses efficiently.

Relevant technical knowledge and expertise in network analysis and data processing.

Access to online resources such as documentation, forums, and tutorials to troubleshoot issues and seek guidance when needed.

Wireshark is a good option for this project due to the following reasons:

1. Wireshark is an open-source and free packet analyzer, which means that the project can be implemented without any significant financial investment.

2. Wireshark provides real-time monitoring and analysis of network traffic, which enables the project to capture and analyze packets as they flow through the network.

3. Wireshark offers a user-friendly interface that makes it easy to analyze network traffic and detect the operating system, sessions, and open ports on the network.

4. Wireshark supports a wide range of network protocols and services, making it a versatile tool that can be used in various network environments.

5. Wireshark provides detailed information about the packets flowing through the network, which can be used to investigate network traffic and identify potential security issues.

Specifications

- Supports UNIX and Windows platforms.

- Can capture live packet data from a network interface.
- Can open files containing packet data captured by tcpdump/WinDump, Wireshark, and other packet capture programs.
- Can import packets from text files containing hex dumps of packet data.
- Shows detailed protocol information in packets.
- Allows users to save any captured packet data.
- Provides options to export some or all packets in various capture file formats.
- Offers packet filtering based on a variety of criteria.
- Provides a search function for packets based on different criteria.
- Allows users to colorize packet displays using filters.
- Generates several statistics.
- And offers many more features.

Live Capture from Various Network Media

Wireshark can capture network traffic from different types of network media such as Ethernet, Wireless LAN, Bluetooth, USB, and others. However, some hardware and operating systems may limit the supported media types.

Import Files from Different Capture Programs

Wireshark can open packet captures from a broad range of capture programs.

Export Files to Different Capture Programs

Wireshark can save captured packets in various formats that are supported by other capture software.

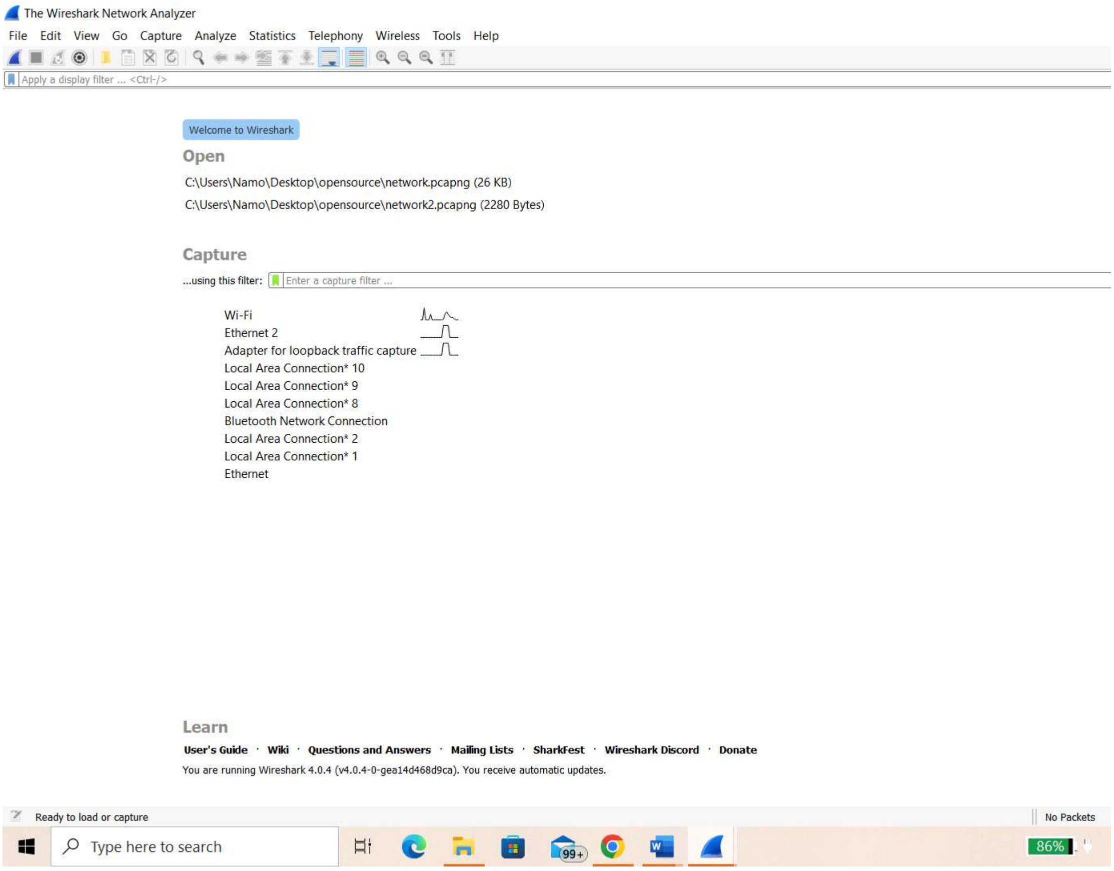
Numerous Protocol Dissectors

Wireshark has protocol dissectors (also known as decoders) for many protocols, which can be found in Appendix C, Protocols and Protocol Fields.

Free and Open-Source Software

Wireshark is a free and open-source software that is licensed under the GNU General Public License (GPL). You can use Wireshark on multiple computers without worrying about license keys or fees. The source code is also available under the GPL, making it easy for people to add new protocols to Wireshark either as plugins or built into the source code.

Main Page of Wireshark


The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Welcome to Wireshark
Type here to search
99+
86%

Open

C:\Users\Namo\Desktop\opensource\network.pcapng (26 KB)

C:\Users\Namo\Desktop\opensource\network2.pcapng (2280 Bytes)

Capture

...using this filter: Enter a capture filter …

Wi-Fi
Ethernet 2
Adapter for loopback traffic capture
Local Area Connection* 10
Local Area Connection* 9
Local Area Connection* 8
Bluetooth Network Connection
Local Area Connection* 2
Local Area Connection* 1
Ethernet

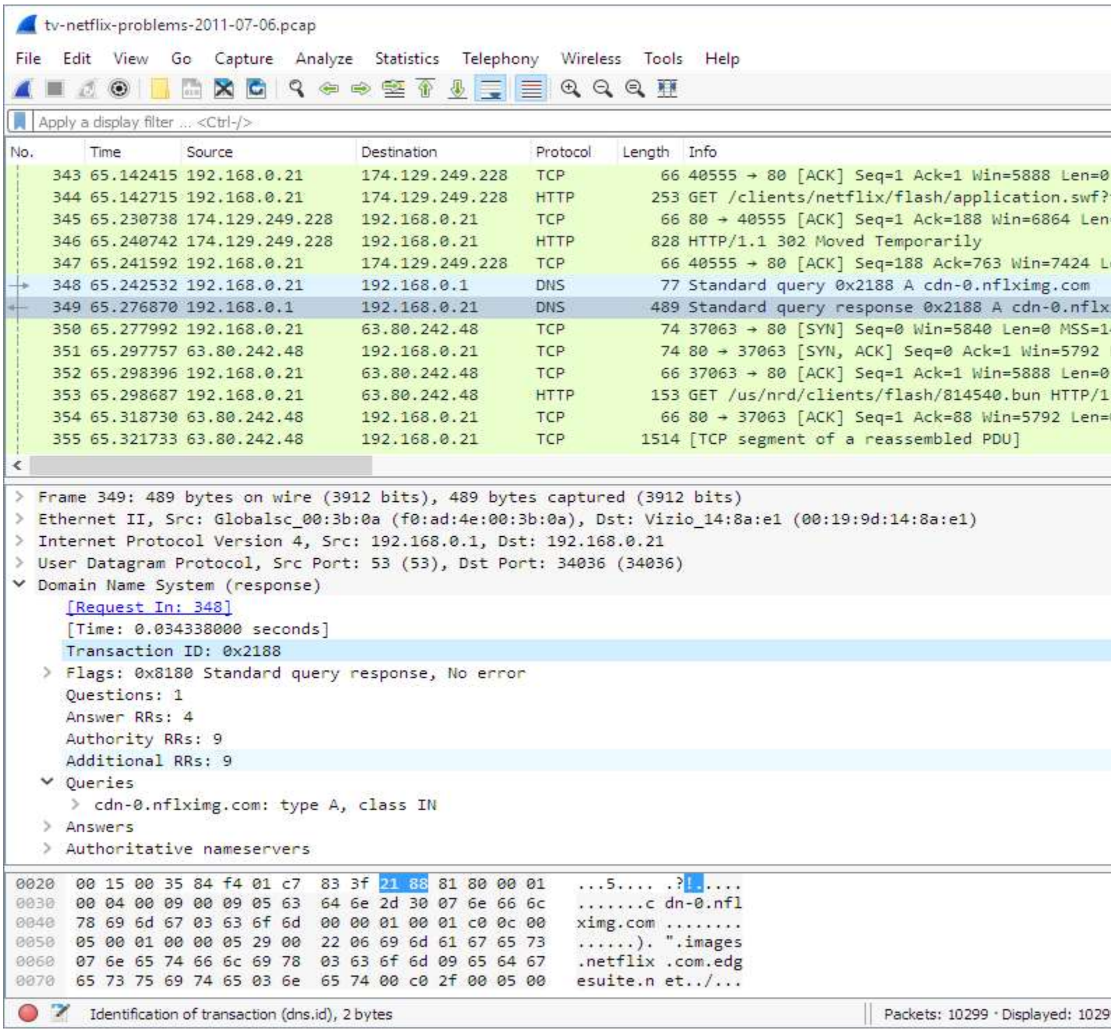
Learn

User's Guide · Wiki · Questions and Answers · Mailing Lists · SharkFest · Wireshark Discord · Donate

You are running Wireshark 4.0.4 (v4.0.4-0-gea14d468d9ca). You receive automatic updates.

Ready to load or capture | No Packets

When we click on Capture Button


tv-netflix-problems-2011-07-06.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.   Time      Source          Destination     Protocol  Length Info
  343 65.142415 192.168.0.21    174.129.249.228 TCP           66 40555 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0
  344 65.142715 192.168.0.21    174.129.249.228 HTTP         253 GET /clients/netflix/flash/application.swf?
  345 65.230738 174.129.249.228 192.168.0.21    TCP           66 80 → 40555 [ACK] Seq=1 Ack=188 Win=6864 Len
  346 65.240742 174.129.249.228 192.168.0.21    HTTP         828 HTTP/1.1 302 Moved Temporarily
  347 65.241592 192.168.0.21    174.129.249.228 TCP           66 40555 → 80 [ACK] Seq=188 Ack=763 Win=7424 L
  348 65.242532 192.168.0.21    192.168.0.1     DNS           77 Standard query 0x2188 A cdn-0.nflximg.com
  349 65.276870 192.168.0.1     192.168.0.21    DNS          489 Standard query response 0x2188 A cdn-0.nflx
  350 65.277992 192.168.0.21    63.80.242.48    TCP           74 37063 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1
  351 65.297757 63.80.242.48    192.168.0.21    TCP           74 80 → 37063 [SYN, ACK] Seq=0 Ack=1 Win=5792
  352 65.298396 192.168.0.21    63.80.242.48    TCP           66 37063 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0
  353 65.298687 192.168.0.21    63.80.242.48    HTTP         153 GET /us/nrd/clients/flash/814540.bun HTTP/1
  354 65.318730 63.80.242.48    192.168.0.21    TCP           66 80 → 37063 [ACK] Seq=1 Ack=88 Win=5792 Len=
  355 65.321733 63.80.242.48    192.168.0.21    TCP         1514 [TCP segment of a reassembled PDU]
```

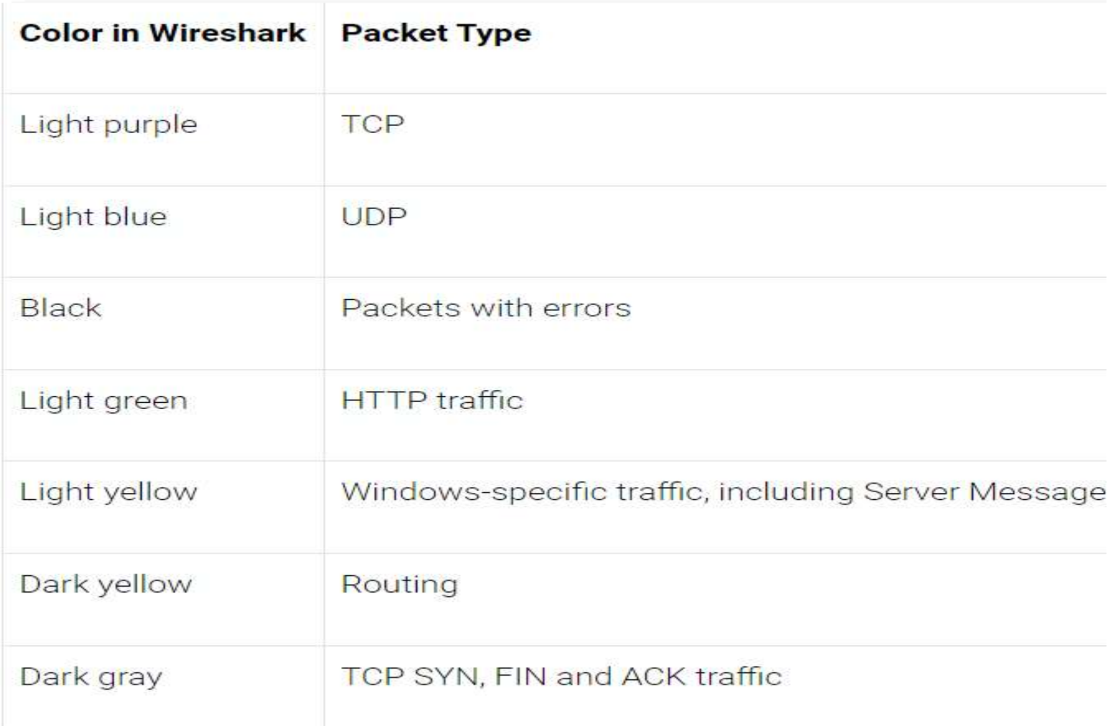
```
> Frame 349: 489 bytes on wire (3912 bits), 489 bytes captured (3912 bits)
> Ethernet II, Src: Globalsc_00:3b:0a (f0:ad:4e:00:3b:0a), Dst: Vizio_14:8a:e1 (00:19:9d:14:8a:e1)
> Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.21
> User Datagram Protocol, Src Port: 53 (53), Dst Port: 34036 (34036)
v Domain Name System (response)
    [Request In: 348]
    [Time: 0.034338000 seconds]
    Transaction ID: 0x2188
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 4
    Authority RRs: 9
    Additional RRs: 9
  v Queries
    > cdn-0.nflximg.com: type A, class IN
  > Answers
  > Authoritative nameservers
```

```
0020  00 15 00 35 84 f4 01 c7  83 3f 21 88 81 80 00 01   ...5.... .?!.....
0030  00 04 00 09 00 09 05 63  64 6e 2d 30 07 6e 66 6c   .......c dn-0.nfl
0040  78 69 6d 67 03 63 6f 6d  00 00 01 00 01 c0 0c 00   ximg.com ........
0050  05 00 01 00 00 05 29 00  22 06 69 6d 61 67 65 73   ......). ".images
0060  07 6e 65 74 66 6c 69 78  03 63 6f 6d 09 65 64 67   .netflix .com.edg
0070  65 73 75 69 74 65 03 6e  65 74 00 c0 2f 00 05 00   esuite.n et../...
```

Identification of transaction (dns.id), 2 bytes

Packets: 10299 · Displayed: 1029

Colour Code for Wireshark



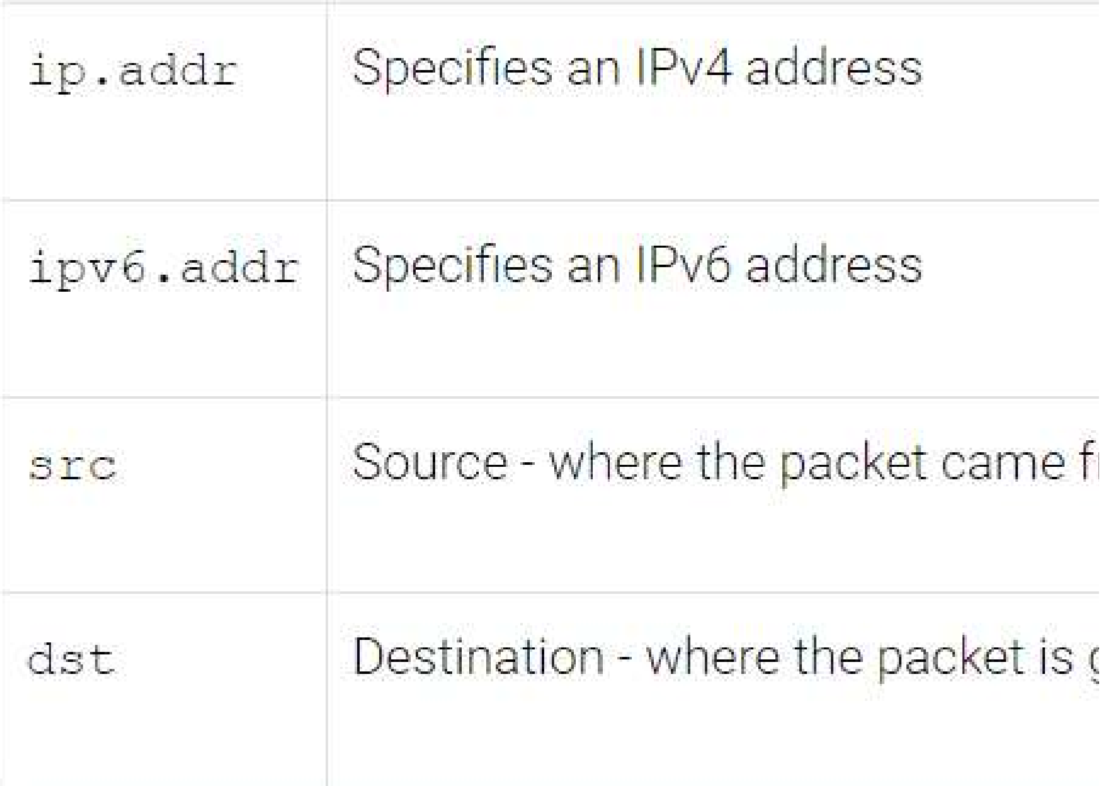
Color in Wireshark	Packet Type
Light purple	TCP
Light blue	UDP
Black	Packets with errors
Light green	HTTP traffic
Light yellow	Windows-specific traffic, including Server Message
Dark yellow	Routing
Dark gray	TCP SYN, FIN and ACK traffic

How to Filter and Inspect Packets in Wireshark

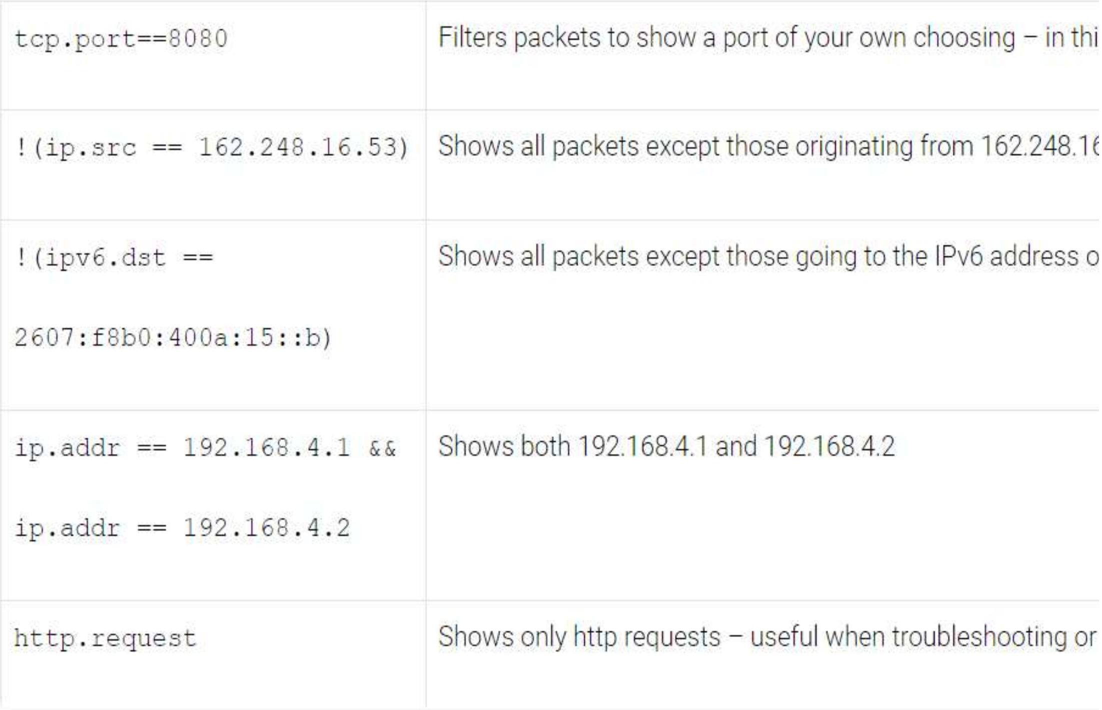
You can apply Wireshark filters in two ways:

- In the Display Filter window, at the top of the screen
- By highlighting a packet (or a portion of a packet) and right-clicking on the packet

Wireshark filters use key phrases, such as the following:



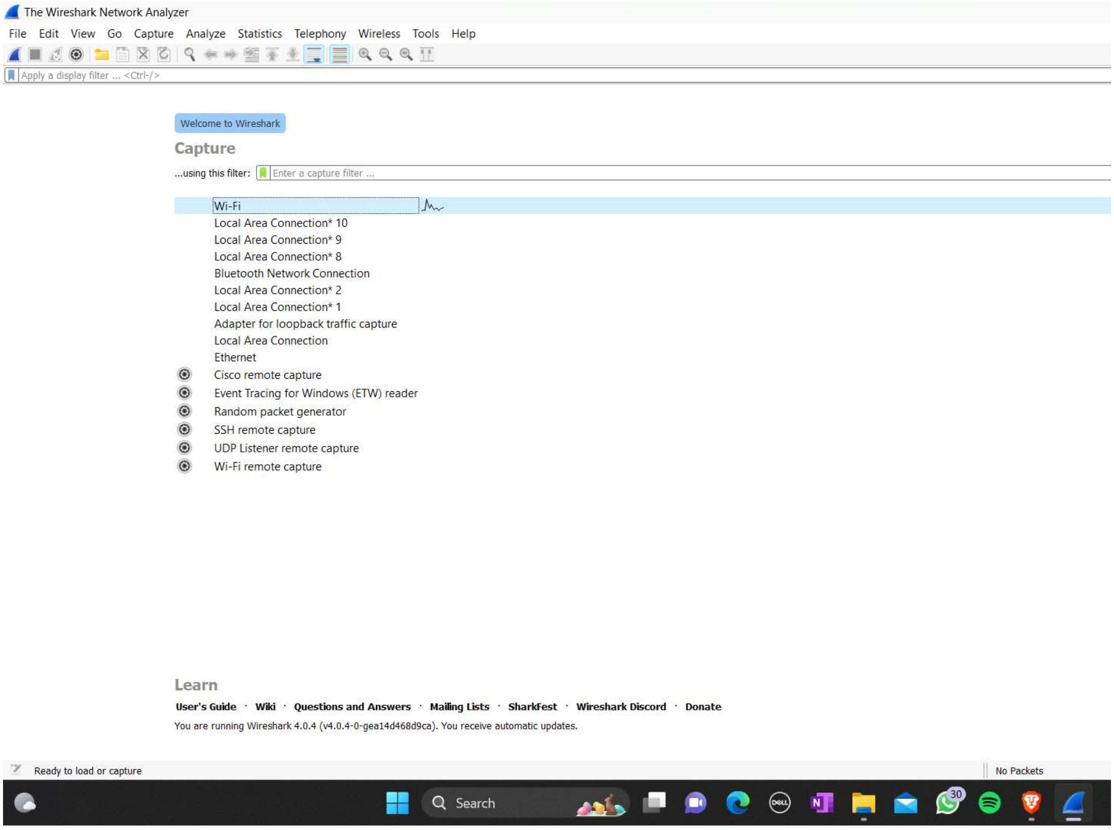
ip.addr	Specifies an IPv4 address
ipv6.addr	Specifies an IPv6 address
src	Source - where the packet came f
dst	Destination - where the packet is



tcp.port==8080	Filters packets to show a port of your own choosing – in thi
!(ip.src == 162.248.16.53)	Shows all packets except those originating from 162.248.1
!(ipv6.dst == 2607:f8b0:400a:15::b)	Shows all packets except those going to the IPv6 address o
ip.addr == 192.168.4.1 && ip.addr == 192.168.4.2	Shows both 192.168.4.1 and 192.168.4.2
http.request	Shows only http requests – useful when troubleshooting or

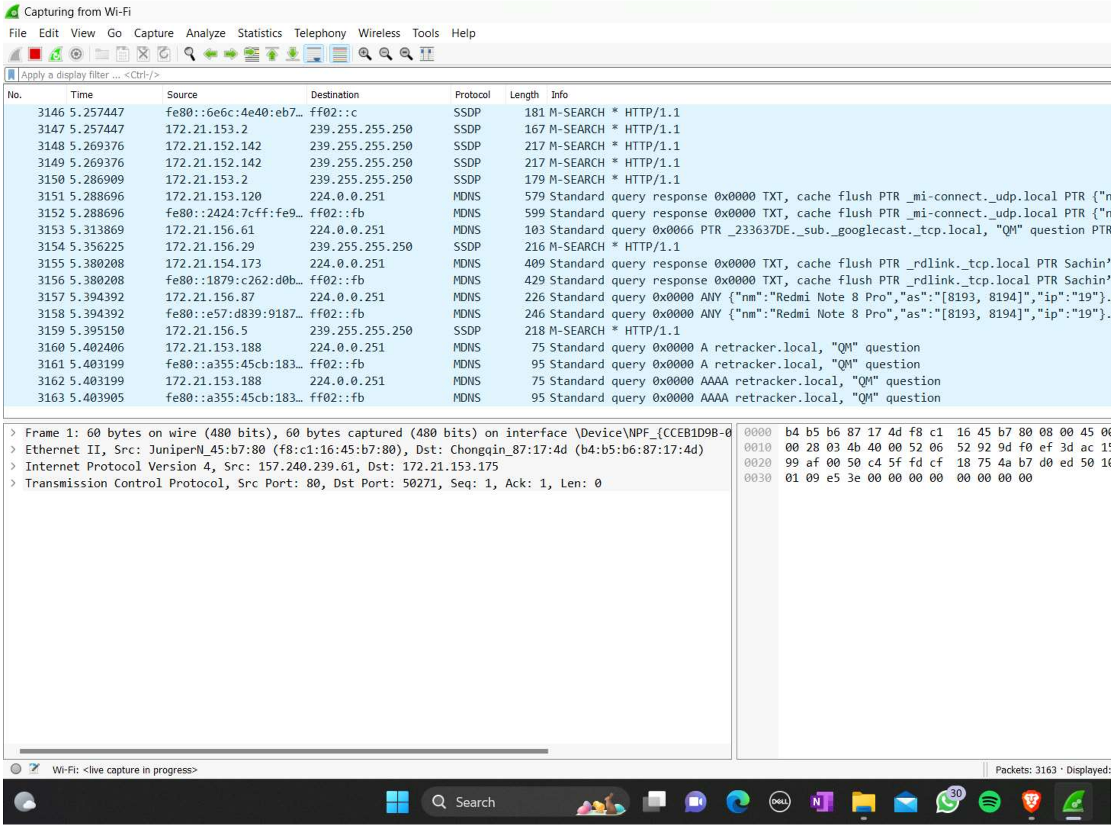
Finding the operating system

- Start Wireshark: Open Wireshark on your computer.


The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Welcome to Wireshark
Capture
…using this filter:
Enter a capture filter …
Wi-Fi
Local Area Connection* 10
Local Area Connection* 9
Local Area Connection* 8
Bluetooth Network Connection
Local Area Connection* 2
Local Area Connection* 1
Adapter for loopback traffic capture
Local Area Connection
Ethernet
Cisco remote capture
Event Tracing for Windows (ETW) reader
Random packet generator
SSH remote capture
UDP Listener remote capture
Wi-Fi remote capture
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists · SharkFest · Wireshark Discord · Donate
Ready to load or capture
No Packets
Search
30

You are running Wireshark 4.0.4 (v4.0.4-0-gea14d468d9ca). You receive automatic updates.

- Capture network traffic: To analyze network traffic, you need to capture it first. In Wireshark, you can select the appropriate network interface to capture the traffic you want to analyze. Click on the "Capture" menu and then select "Interfaces". Select the interface you want to capture traffic from, and click on "Start" to begin capturing packets.



Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

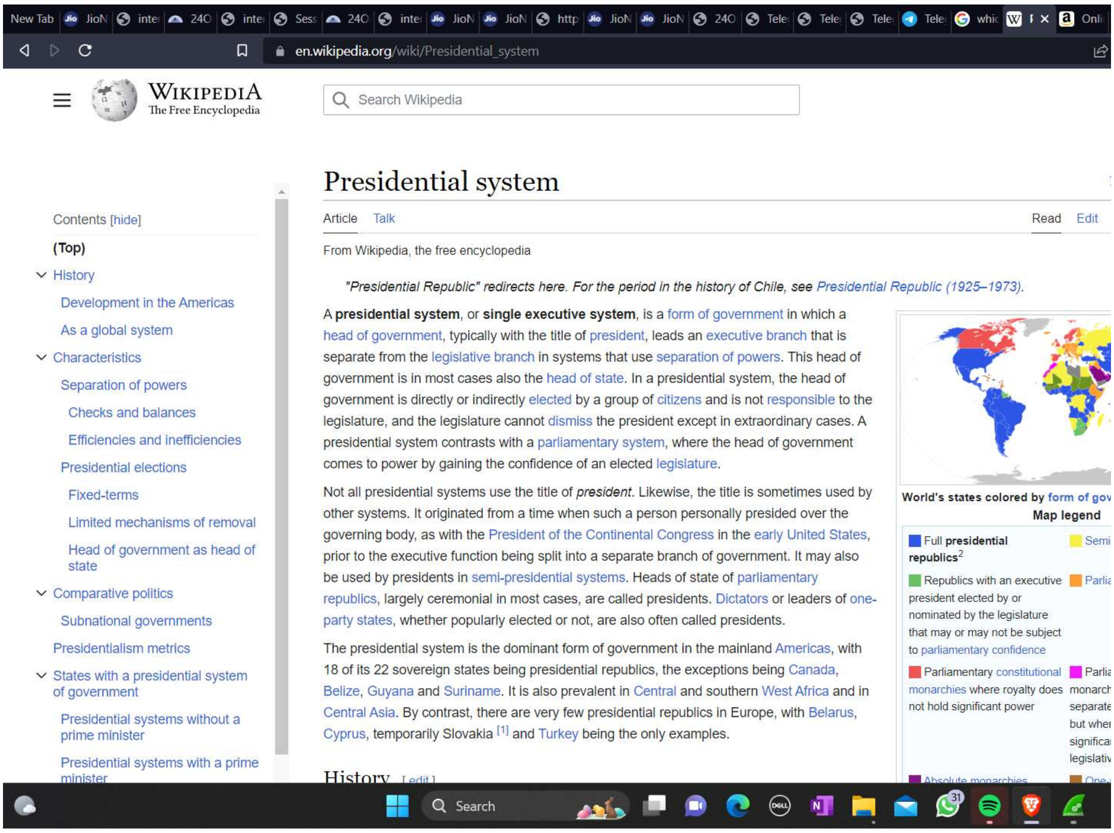
```
No.    Time      Source                 Destination       Protocol  Length Info
  3146 5.257447  fe80::6e6c:4e40:eb7…   ff02::c           SSDP      181 M-SEARCH * HTTP/1.1
  3147 5.257447  172.21.153.2           239.255.255.250   SSDP      167 M-SEARCH * HTTP/1.1
  3148 5.269376  172.21.152.142         239.255.255.250   SSDP      217 M-SEARCH * HTTP/1.1
  3149 5.269376  172.21.152.142         239.255.255.250   SSDP      217 M-SEARCH * HTTP/1.1
  3150 5.286909  172.21.153.2           239.255.255.250   SSDP      179 M-SEARCH * HTTP/1.1
  3151 5.288696  172.21.153.120         224.0.0.251       MDNS      579 Standard query response 0x0000 TXT, cache flush PTR _mi-connect._udp.local PTR {"n
  3152 5.288696  fe80::2424:7cff:fe9…   ff02::fb          MDNS      599 Standard query response 0x0000 TXT, cache flush PTR _mi-connect._udp.local PTR {"n
  3153 5.313869  172.21.156.61          224.0.0.251       MDNS      103 Standard query 0x0066 PTR _233637DE._sub._googlecast._tcp.local, "QM" question PTR
  3154 5.356225  172.21.156.29          239.255.255.250   SSDP      216 M-SEARCH * HTTP/1.1
  3155 5.380208  172.21.154.173         224.0.0.251       MDNS      409 Standard query response 0x0000 TXT, cache flush PTR _rdlink._tcp.local PTR Sachin'
  3156 5.380208  fe80::1879:c262:d0b…   ff02::fb          MDNS      429 Standard query response 0x0000 TXT, cache flush PTR _rdlink._tcp.local PTR Sachin'
  3157 5.394392  172.21.156.87          224.0.0.251       MDNS      226 Standard query 0x0000 ANY {"nm":"Redmi Note 8 Pro","as":"[8193, 8194]","ip":"19"}.
  3158 5.394392  fe80::e57:d839:9187…   ff02::fb          MDNS      246 Standard query 0x0000 ANY {"nm":"Redmi Note 8 Pro","as":"[8193, 8194]","ip":"19"}.
  3159 5.395150  172.21.156.5           239.255.255.250   SSDP      218 M-SEARCH * HTTP/1.1
  3160 5.402406  172.21.153.188         224.0.0.251       MDNS       75 Standard query 0x0000 A retracker.local, "QM" question
  3161 5.403199  fe80::a355:45cb:183…   ff02::fb          MDNS       95 Standard query 0x0000 A retracker.local, "QM" question
  3162 5.403199  172.21.153.188         224.0.0.251       MDNS       75 Standard query 0x0000 AAAA retracker.local, "QM" question
  3163 5.403905  fe80::a355:45cb:183…   ff02::fb          MDNS       95 Standard query 0x0000 AAAA retracker.local, "QM" question
```

```
> Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{CCEB1D9B-0
> Ethernet II, Src: JuniperN_45:b7:80 (f8:c1:16:45:b7:80), Dst: Chongqin_87:17:4d (b4:b5:b6:87:17:4d)
> Internet Protocol Version 4, Src: 157.240.239.61, Dst: 172.21.153.175
> Transmission Control Protocol, Src Port: 80, Dst Port: 50271, Seq: 1, Ack: 1, Len: 0
```

```
0000  b4 b5 b6 87 17 4d f8 c1  16 45 b7 80 08 00 45 0
0010  00 28 03 4b 40 00 52 06  52 92 9d f0 ef 3d ac 1
0020  99 af 00 50 c4 5f fd cf  18 75 4a b7 d0 ed 50 1
0030  01 09 e5 3e 00 00 00 00  00 00 00 00
```

```
Wi-Fi: <live capture in progress>        Packets: 3163 · Displayed:
```

- Filter packets: With Wireshark, you can apply filters to reduce the amount of data you need to analyze. To filter the packets by the TTL field, use the following filter expression: "ip.ttl==128". This filter expression will show only the packets with a TTL of 128, which is commonly used by certain operating systems.



Contents [hide]

(Top)

- History
  - Development in the Americas
  - As a global system
- Characteristics
  - Separation of powers
    - Checks and balances
    - Efficiencies and inefficiencies
  - Presidential elections
    - Fixed-terms
    - Limited mechanisms of removal
    - Head of government as head of state
- Comparative politics
  - Subnational governments
- Presidentialism metrics
- States with a presidential system of government
  - Presidential systems without a prime minister
  - Presidential systems with a prime minister

Presidential system

Article Talk Read Edit

From Wikipedia, the free encyclopedia

"Presidential Republic" redirects here. For the period in the history of Chile, see Presidential Republic (1925–1973).

A **presidential system**, or **single executive system**, is a form of government in which a head of government, typically with the title of president, leads an executive branch that is separate from the legislative branch in systems that use separation of powers. This head of government is in most cases also the head of state. In a presidential system, the head of government is directly or indirectly elected by a group of citizens and is not responsible to the legislature, and the legislature cannot dismiss the president except in extraordinary cases. A presidential system contrasts with a parliamentary system, where the head of government comes to power by gaining the confidence of an elected legislature.

Not all presidential systems use the title of *president*. Likewise, the title is sometimes used by other systems. It originated from a time when such a person personally presided over the governing body, as with the President of the Continental Congress in the early United States, prior to the executive function being split into a separate branch of government. It may also be used by presidents in semi-presidential systems. Heads of state of parliamentary republics, largely ceremonial in most cases, are called presidents. Dictators or leaders of one-party states, whether popularly elected or not, are also often called presidents.

The presidential system is the dominant form of government in the mainland Americas, with 18 of its 22 sovereign states being presidential republics, the exceptions being Canada, Belize, Guyana and Suriname. It is also prevalent in Central and southern West Africa and in Central Asia. By contrast, there are very few presidential republics in Europe, with Belarus, Cyprus, temporarily Slovakia [1] and Turkey being the only examples.

World's states colored by form of gov

Map legend

🟦 Full **presidential republics**[2]

🟨 Semi

🟩 Republics with an executive president elected by or nominated by the legislature that may or may not be subject to parliamentary confidence

🟧 Parlia

🟥 Parliamentary constitutional monarchies where royalty does not hold significant power

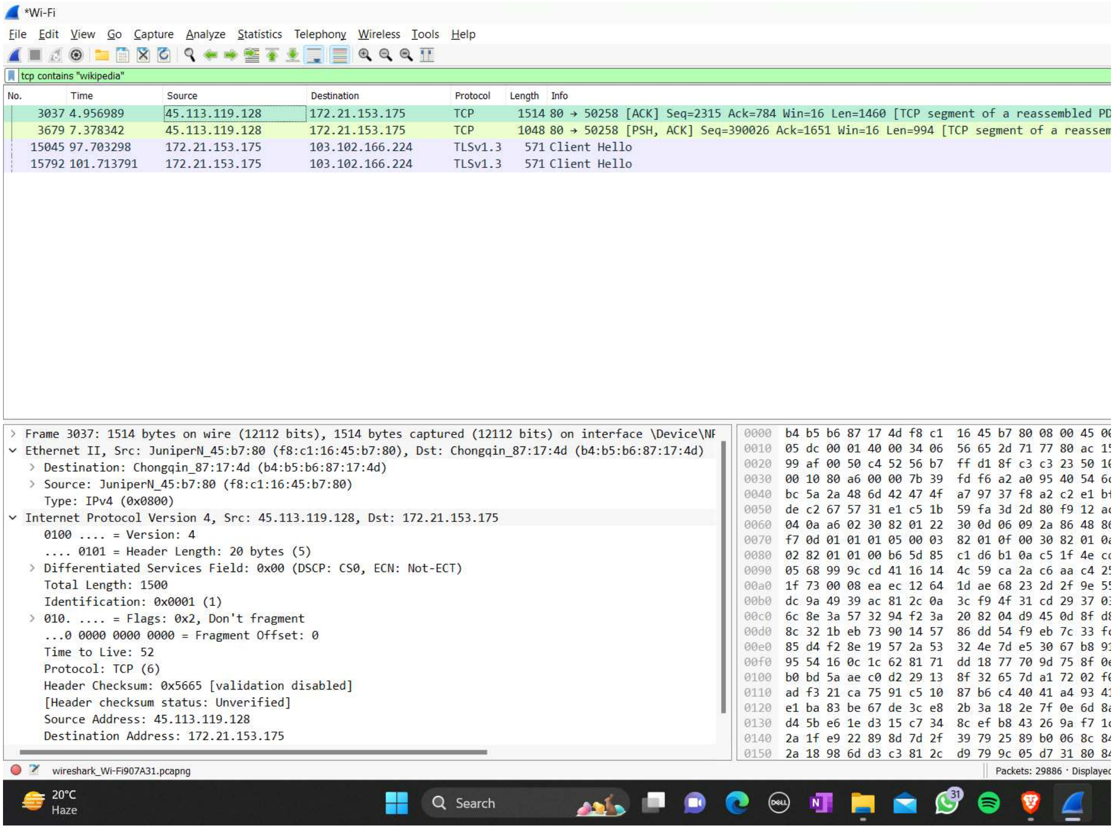
🟪 Parlia monarch separate but wher significa legislativ

Absolute monarchies

One-

History [edit]

- Observe TTL values: Examine the TTL values of the packets that are displayed in the capture window. The TTL values are displayed in the Time to Live column. TTL is a field in the IP header of packets that specifies how many routers the packet can pass through before being discarded.



```
tcp contains "wikipedia"

No.     Time        Source           Destination       Protocol  Length Info
   3037 4.956989    45.113.119.128   172.21.153.175    TCP       1514 80 → 50258 [ACK] Seq=2315 Ack=784 Win=16 Len=1460 [TCP segment of a reassembled PD
   3679 7.378342    45.113.119.128   172.21.153.175    TCP       1048 80 → 50258 [PSH, ACK] Seq=390026 Ack=1651 Win=16 Len=994 [TCP segment of a reassem
  15045 97.703298   172.21.153.175   103.102.166.224   TLSv1.3    571 Client Hello
  15792 101.713791  172.21.153.175   103.102.166.224   TLSv1.3    571 Client Hello
```

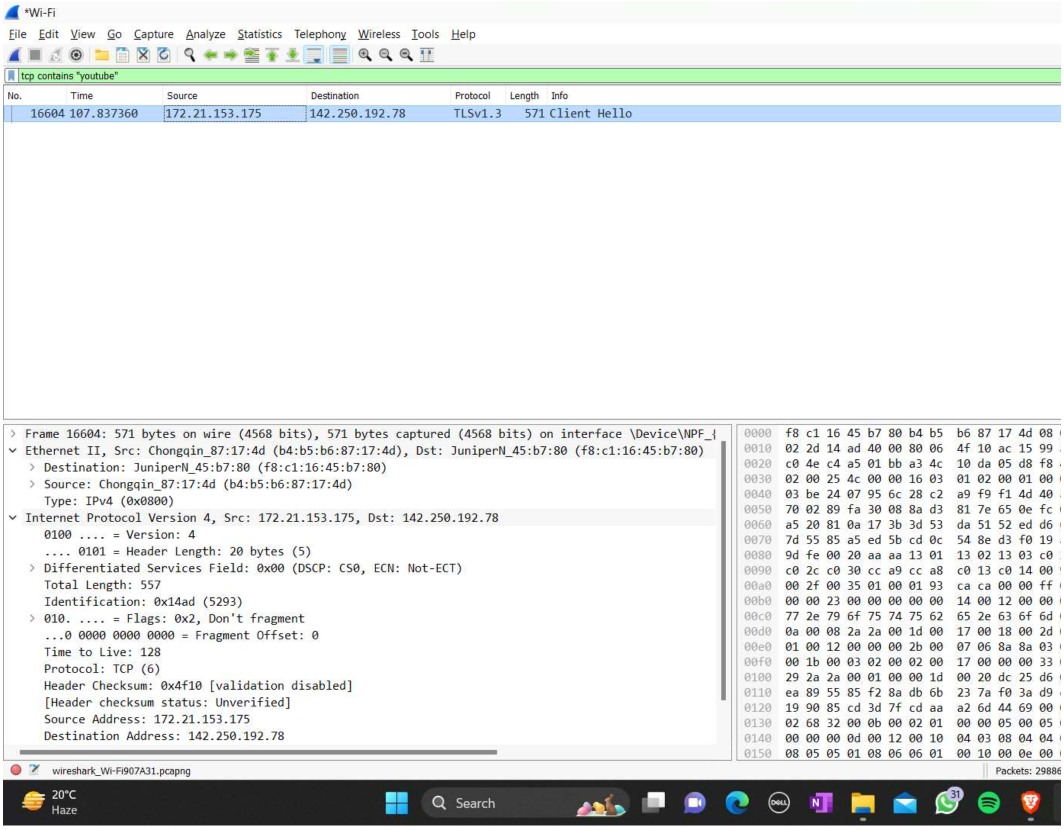
```
> Frame 3037: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface \Device\NF
v Ethernet II, Src: JuniperN_45:b7:80 (f8:c1:16:45:b7:80), Dst: Chongqin_87:17:4d (b4:b5:b6:87:17:4d)
  > Destination: Chongqin_87:17:4d (b4:b5:b6:87:17:4d)
  > Source: JuniperN_45:b7:80 (f8:c1:16:45:b7:80)
    Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 45.113.119.128, Dst: 172.21.153.175
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1500
    Identification: 0x0001 (1)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 52
    Protocol: TCP (6)
    Header Checksum: 0x5665 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 45.113.119.128
    Destination Address: 172.21.153.175
```

```
0000  b4 b5 b6 87 17 4d f8 c1  16 45 b7 80 08 00 45 0
0010  05 dc 00 01 40 00 34 06  56 65 2d 71 77 80 ac 1
0020  99 af 00 50 c4 52 56 b7  ff d1 8f c3 c3 23 50 1
0030  00 10 80 a6 00 00 7b 39  fd f6 a2 a0 95 40 54 6
0040  bc 5a 2a 48 6d 42 47 4f  a7 97 37 f8 a2 c2 e1 b
0050  de c2 67 57 31 e1 c5 1b  59 fa 3d 2d 80 f9 12 a
0060  04 0a a6 02 30 82 01 22  30 0d 06 09 2a 86 48 8
0070  f7 0d 01 01 01 05 00 03  82 01 0f 00 30 82 01 0
0080  02 82 01 01 00 b6 5d 85  c1 d6 b1 0a c5 1f 4e c
0090  05 68 99 9c cd 41 16 14  4c 59 ca 2a c6 aa c4 2
00a0  1f 73 00 08 ea ec 12 64  1d ae 68 23 2d 2f 9e 5
00b0  dc 9a 49 39 ac 81 2c 0a  3c f9 4f 31 cd 29 37 0
00c0  6c 8e 3a 57 32 94 f2 3a  20 82 04 d9 45 0d 8f d
00d0  8c 32 1b eb 73 90 14 57  86 dd 54 f9 eb 7c 33 f
00e0  85 d4 f2 8e 19 57 2a 53  32 4e 7d e5 30 67 b8 9
00f0  95 54 16 0c 1c 62 81 71  dd 18 77 70 9d 75 8f 0
0100  b0 bd 5a ae c0 d2 29 13  8f 32 65 7d a1 72 02 f
0110  ad f3 21 ca 75 91 c5 10  87 b6 c4 40 41 a4 93 4
0120  e1 ba 83 be 67 de 3c e8  2b 3a 18 2e 7f 0e 6d 8
0130  d4 5b e6 1e d3 15 c7 34  8c ef b8 43 26 9a f7 1
0140  2a 1f e9 22 89 8d 7d 2f  39 79 25 89 b0 06 8c 8
0150  2a 18 98 6d d3 c3 81 2c  d9 79 9c 05 d7 31 80 8
```

wireshark_Wi-Fi907A31.pcapng

Packets: 29886 · Displayed

- Determine the operating system: Compare the TTL values of the packets to the TTL values of known operating systems. Different operating systems use different TTL values. For example, Windows typically uses a TTL value of 128, while Linux typically uses a TTL value of 64. By comparing the TTL values of the packets to known values, you can determine the operating system.
- Repeat for other packets: Repeat the process for other packets to get a better idea of the operating systems being used on the network. It is important to analyze a large number of packets to get a more accurate picture of the operating systems being used.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
20°C
Haze
Search
31

tcp contains "youtube"

```
No.    Time        Source          Destination     Protocol  Length Info
16604  107.837360  172.21.153.175  142.250.192.78  TLSv1.3   571    Client Hello
```

```
> Frame 16604: 571 bytes on wire (4568 bits), 571 bytes captured (4568 bits) on interface \Device\NPF_{
v Ethernet II, Src: Chongqin_87:17:4d (b4:b5:b6:87:17:4d), Dst: JuniperN_45:b7:80 (f8:c1:16:45:b7:80)
  > Destination: JuniperN_45:b7:80 (f8:c1:16:45:b7:80)
  > Source: Chongqin_87:17:4d (b4:b5:b6:87:17:4d)
    Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 172.21.153.175, Dst: 142.250.192.78
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 557
    Identification: 0x14ad (5293)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: TCP (6)
    Header Checksum: 0x4f10 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.21.153.175
    Destination Address: 142.250.192.78
```

```
0000  f8 c1 16 45 b7 80 b4 b5  b6 87 17 4d 08
0010  02 2d 14 ad 40 00 80 06  4f 10 ac 15 99
0020  c0 4e c4 a5 01 bb a3 4c  10 da 05 d8 f8
0030  02 00 25 4c 00 00 16 03  01 02 00 01 00
0040  03 be 24 07 95 6c 28 c2  a9 f9 f1 4d 40
0050  70 02 89 fa 30 08 8a d3  81 7e 65 0e fc
0060  a5 20 81 0a 17 3b 3d 53  da 51 52 ed d6
0070  7d 55 85 a5 ed 5b cd 0c  54 8e d3 f0 19
0080  9d fe 00 20 aa aa 13 01  13 02 13 03 c0
0090  c0 2c c0 30 cc a9 cc a8  c0 13 c0 14 00
00a0  00 2f 00 35 01 00 01 93  ca ca 00 00 ff
00b0  00 00 23 00 00 00 00 00  14 00 12 00 00
00c0  77 2e 79 6f 75 74 75 62  65 2e 63 6f 6d
00d0  0a 00 08 2a 2a 00 1d 00  17 00 18 00 2d
00e0  01 00 12 00 00 00 2b 00  07 06 8a 8a 03
00f0  00 1b 00 03 02 00 02 00  17 00 00 00 33
0100  29 2a 2a 00 01 00 00 1d  00 20 dc 25 d6
0110  ea 89 55 85 f2 8a db 6b  23 7a f0 3a d9
0120  19 90 85 cd 3d 7f cd aa  a2 6d 44 69 00
0130  02 68 32 00 0b 00 02 01  00 00 05 00 05
0140  00 00 00 0d 00 12 00 10  04 03 08 04 04
0150  08 05 05 01 08 06 06 01  00 10 00 0e 00
```

wireshark_Wi-Fi907A31.pcapng

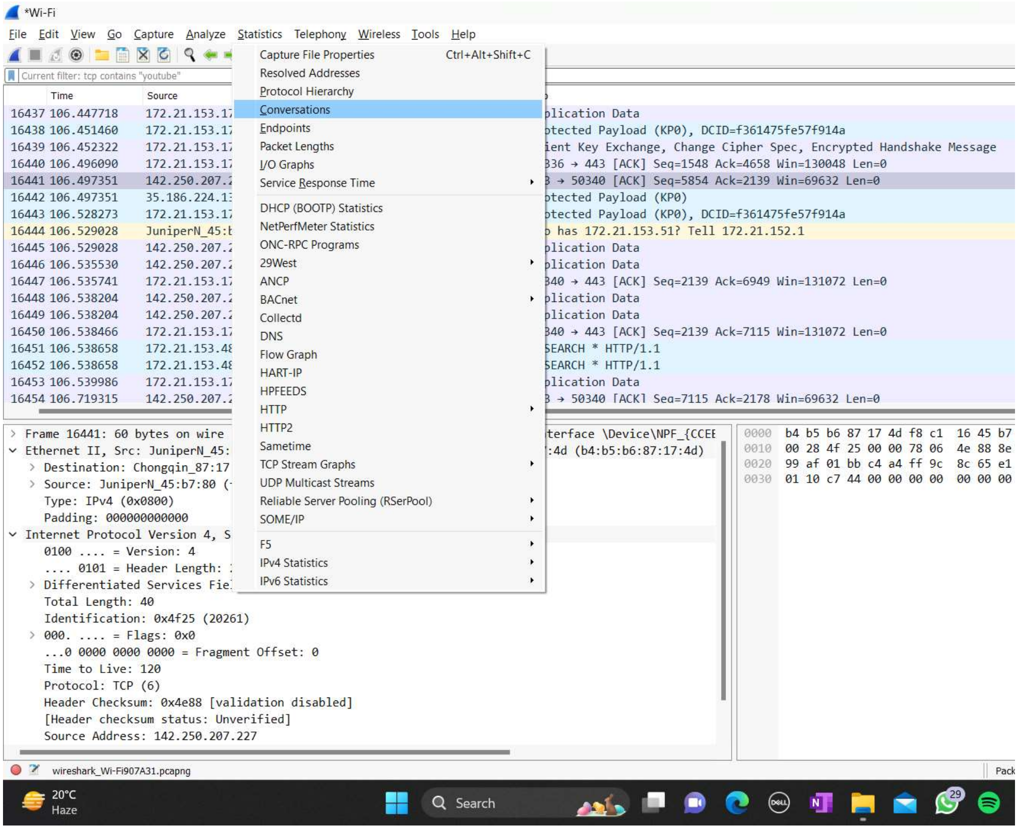
Packets: 29886

In summary, analyzing the TTL values of packets in Wireshark can provide valuable information about the operating systems being used on a network. By following the steps outlined above, you can use Wireshark to find the operating system being used by analyzing the TTL values of captured packets.

Finding TCP Session

The Time to Live (TTL) field in the IP header is a useful tool in identifying TCP sessions in Wireshark. Here's a more detailed step-by-step procedure to find TCP sessions using TTL:

- Open Wireshark and load the capture file containing the network traffic you want to analyze.
- Filter the traffic to display only TCP packets. You can do this by typing "tcp" in the filter box or by going to "Analyse" -> "Display Filters" -> "tcp" and clicking "Apply".
- Look for packets with different TTL values. The TTL field in the IP header specifies the maximum number of hops a packet can take before being discarded. Each operating system sets a default TTL value for outgoing packets, which can be used to identify the operating system.



Current filter: tcp contains "youtube"

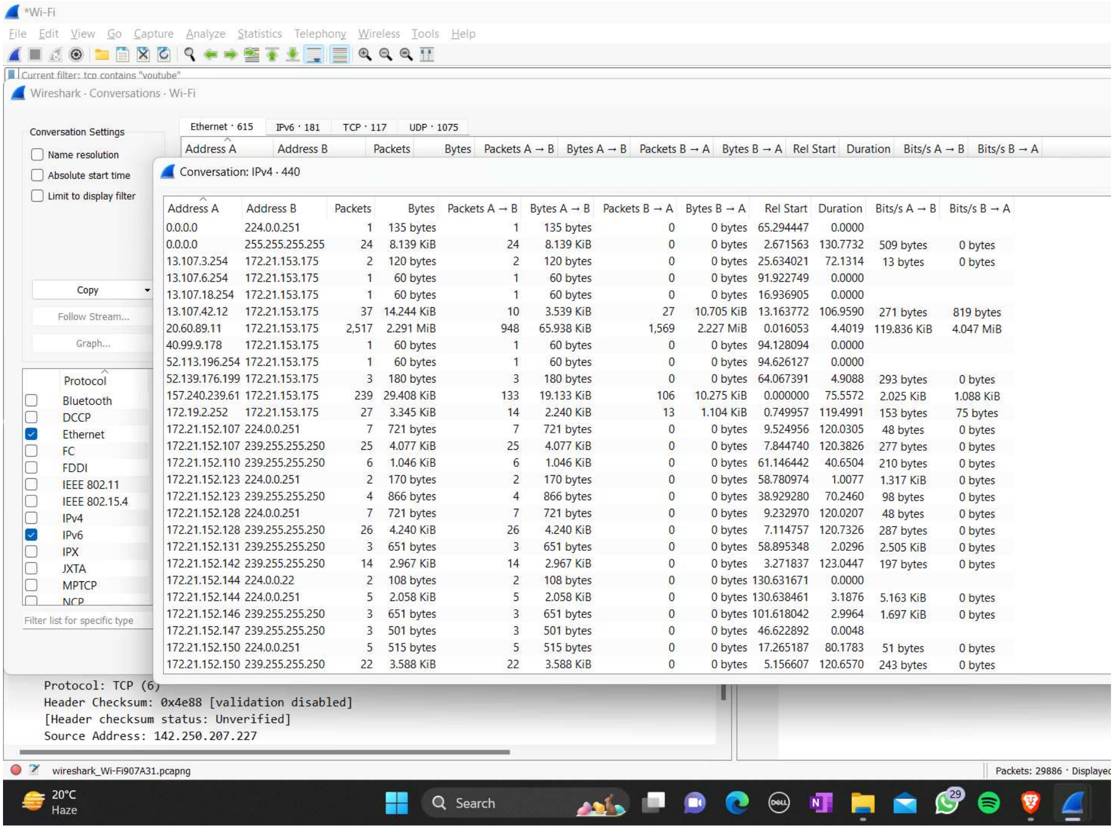
```
      Time          Source
16437 106.447718    172.21.153.17      plication Data
16438 106.451460    172.21.153.17      otected Payload (KP0), DCID=f361475fe57f914a
16439 106.452322    172.21.153.17      ient Key Exchange, Change Cipher Spec, Encrypted Handshake Message
16440 106.496090    172.21.153.17      336 → 443 [ACK] Seq=1548 Ack=4658 Win=130048 Len=0
16441 106.497351    142.250.207.2      3 → 50340 [ACK] Seq=5854 Ack=2139 Win=69632 Len=0
16442 106.497351    35.186.224.13      otected Payload (KP0)
16443 106.528273    172.21.153.17      otected Payload (KP0), DCID=f361475fe57f914a
16444 106.529028    JuniperN_45:b      o has 172.21.153.51? Tell 172.21.152.1
16445 106.529028    142.250.207.2      plication Data
16446 106.535530    142.250.207.2      plication Data
16447 106.535741    172.21.153.17      340 → 443 [ACK] Seq=2139 Ack=6949 Win=131072 Len=0
16448 106.538204    142.250.207.2      plication Data
16449 106.538204    142.250.207.2      plication Data
16450 106.538466    172.21.153.17      340 → 443 [ACK] Seq=2139 Ack=7115 Win=131072 Len=0
16451 106.538658    172.21.153.48      SEARCH * HTTP/1.1
16452 106.538658    172.21.153.48      SEARCH * HTTP/1.1
16453 106.539986    172.21.153.17      plication Data
16454 106.719315    142.250.207.2      3 → 50340 [ACK] Seq=7115 Ack=2178 Win=69632 Len=0
```

```
> Frame 16441: 60 bytes on wire              terface \Device\NPF_{CCEE
v Ethernet II, Src: JuniperN_45:             :4d (b4:b5:b6:87:17:4d)
  > Destination: Chongqin_87:17
  > Source: JuniperN_45:b7:80 (
    Type: IPv4 (0x0800)
    Padding: 000000000000
v Internet Protocol Version 4, S
    0100 .... = Version: 4
    .... 0101 = Header Length: 
  > Differentiated Services Fie
    Total Length: 40
    Identification: 0x4f25 (20261)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 120
    Protocol: TCP (6)
    Header Checksum: 0x4e88 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 142.250.207.227
```

```
0000  b4 b5 b6 87 17 4d f8 c1  16 45 b7
0010  00 28 4f 25 00 00 78 06  4e 88 8e
0020  99 af 01 bb c4 a4 ff 9c  8c 65 e1
0030  01 10 c7 44 00 00 00 00  00 00 00
```

wireshark_Wi-Fi907A31.pcapng

- To view the TTL field, look for the "Time to Live" column in the packet list. If this column is not visible, right-click on any of the column headers and select "Time to Live" from the drop-down menu.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Wireshark · Conversations · Wi-Fi
Ethernet · 615
IPv6 · 181
TCP · 117
UDP · 1075
Copy
Follow Stream...
Graph...
Filter list for specific type
Search
20°C
Haze

Current filter: tcp contains "youtube"

Conversation Settings

☐ Name resolution

☐ Absolute start time

☐ Limit to display filter

Protocol

☐ Bluetooth
☐ DCCP
☑ Ethernet
☐ FC
☐ FDDI
☐ IEEE 802.11
☐ IEEE 802.15.4
☐ IPv4
☑ IPv6
☐ IPX
☐ JXTA
☐ MPTCP
☐ NCP

Conversation: IPv4 · 440

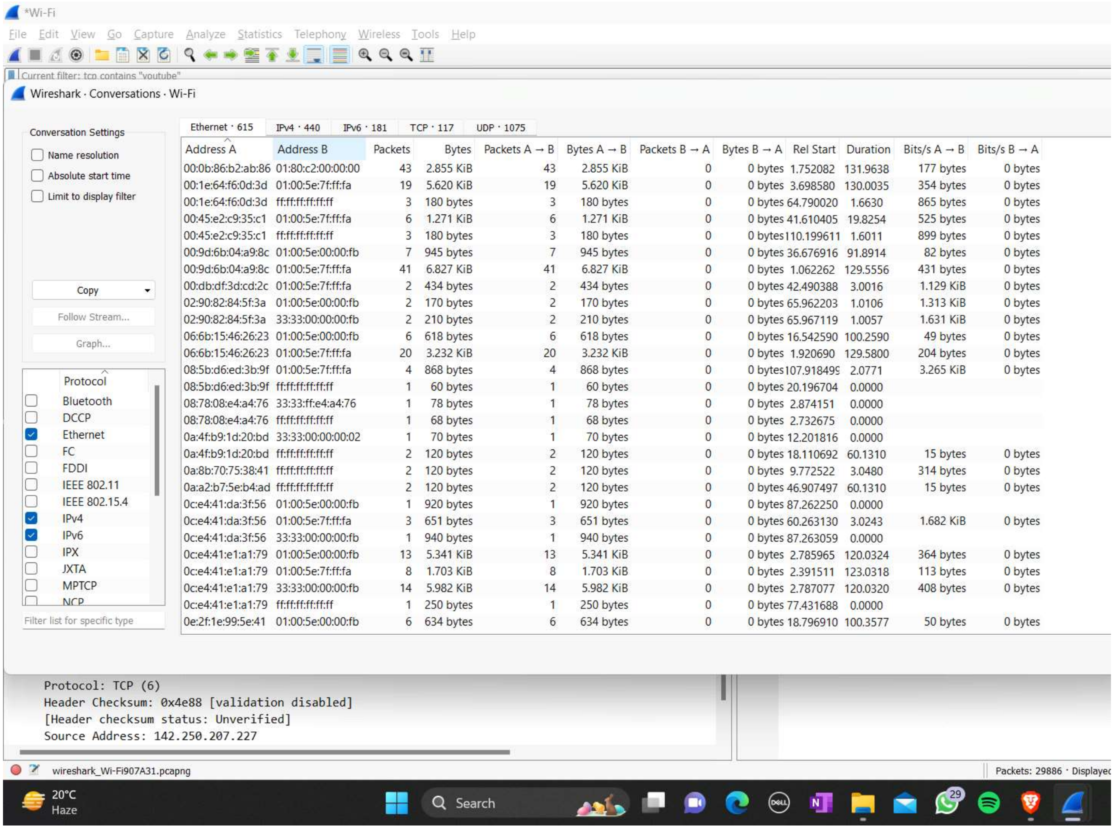
Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
0.0.0.0	224.0.0.251	1	135 bytes	1	135 bytes	0	0 bytes	65.294447	0.0000		
0.0.0.0	255.255.255.255	24	8.139 KiB	24	8.139 KiB	0	0 bytes	2.671563	130.7732	509 bytes	0 bytes
13.107.3.254	172.21.153.175	2	120 bytes	2	120 bytes	0	0 bytes	25.634021	72.1314	13 bytes	0 bytes
13.107.6.254	172.21.153.175	1	60 bytes	1	60 bytes	0	0 bytes	91.922749	0.0000		
13.107.18.254	172.21.153.175	1	60 bytes	1	60 bytes	0	0 bytes	16.936905	0.0000		
13.107.42.12	172.21.153.175	37	14.244 KiB	10	3.539 KiB	27	10.705 KiB	13.163772	106.9590	271 bytes	819 bytes
20.60.89.11	172.21.153.175	2,517	2.291 MiB	948	65.938 KiB	1,569	2.227 MiB	0.016053	4.4019	119.836 KiB	4.047 MiB
40.99.9.178	172.21.153.175	1	60 bytes	1	60 bytes	0	0 bytes	94.128094	0.0000		
52.113.196.254	172.21.153.175	1	60 bytes	1	60 bytes	0	0 bytes	94.626127	0.0000		
52.139.176.199	172.21.153.175	3	180 bytes	3	180 bytes	0	0 bytes	64.067391	4.9088	293 bytes	0 bytes
157.240.239.61	172.21.153.175	239	29.408 KiB	133	19.133 KiB	106	10.275 KiB	0.000000	75.5572	2.025 KiB	1.088 KiB
172.19.2.252	172.21.153.175	27	3.345 KiB	14	2.240 KiB	13	1.104 KiB	0.749957	119.4991	153 bytes	75 bytes
172.21.152.107	224.0.0.251	7	721 bytes	7	721 bytes	0	0 bytes	9.524956	120.0305	48 bytes	0 bytes
172.21.152.107	239.255.255.250	25	4.077 KiB	25	4.077 KiB	0	0 bytes	7.844740	120.3826	277 bytes	0 bytes
172.21.152.110	239.255.255.250	6	1.046 KiB	6	1.046 KiB	0	0 bytes	61.146442	40.6504	210 bytes	0 bytes
172.21.152.123	224.0.0.251	2	170 bytes	2	170 bytes	0	0 bytes	58.780974	1.0077	1.317 KiB	0 bytes
172.21.152.123	239.255.255.250	4	866 bytes	4	866 bytes	0	0 bytes	38.929280	70.2460	98 bytes	0 bytes
172.21.152.128	224.0.0.251	7	721 bytes	7	721 bytes	0	0 bytes	9.232970	120.0207	48 bytes	0 bytes
172.21.152.128	239.255.255.250	26	4.240 KiB	26	4.240 KiB	0	0 bytes	7.114757	120.7326	287 bytes	0 bytes
172.21.152.131	239.255.255.250	3	651 bytes	3	651 bytes	0	0 bytes	58.895348	2.0296	2.505 KiB	0 bytes
172.21.152.142	239.255.255.250	14	2.967 KiB	14	2.967 KiB	0	0 bytes	3.271837	123.0447	197 bytes	0 bytes
172.21.152.144	224.0.0.22	2	108 bytes	2	108 bytes	0	0 bytes	130.631671	0.0000		
172.21.152.144	224.0.0.251	5	2.058 KiB	5	2.058 KiB	0	0 bytes	130.638461	3.1876	5.163 KiB	0 bytes
172.21.152.146	239.255.255.250	3	651 bytes	3	651 bytes	0	0 bytes	101.618042	2.9964	1.697 KiB	0 bytes
172.21.152.147	239.255.255.250	3	501 bytes	3	501 bytes	0	0 bytes	46.622892	0.0048		
172.21.152.150	224.0.0.251	5	515 bytes	5	515 bytes	0	0 bytes	17.265187	80.1783	51 bytes	0 bytes
172.21.152.150	239.255.255.250	22	3.588 KiB	22	3.588 KiB	0	0 bytes	5.156607	120.6570	243 bytes	0 bytes

```
Protocol: TCP (6)
Header Checksum: 0x4e88 [validation disabled]
[Header checksum status: Unverified]
Source Address: 142.250.207.227
```

wireshark_Wi-Fi907A31.pcapng

Packets: 29886 · Displayed

- Select a packet with a TTL value that is different from the others. Right-click on the packet and select "Follow > TCP stream" from the context menu.
- The TCP stream window will show the packets that belong to the selected session. The source and destination IP addresses and port numbers will be displayed at the top of the window.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Wireshark · Conversations · Wi-Fi
Ethernet · 615
IPv4 · 440
IPv6 · 181
TCP · 117
UDP · 1075
Copy
Follow Stream...
Graph...
Filter list for specific type
20°C
Haze
Search

Current filter: tcp contains "youtube"

Conversation Settings

- ☐ Name resolution
- ☐ Absolute start time
- ☐ Limit to display filter

Protocol

- ☐ Bluetooth
- ☐ DCCP
- ☑ Ethernet
- ☐ FC
- ☐ FDDI
- ☐ IEEE 802.11
- ☐ IEEE 802.15.4
- ☑ IPv4
- ☑ IPv6
- ☐ IPX
- ☐ JXTA
- ☐ MPTCP
- ☐ NCP

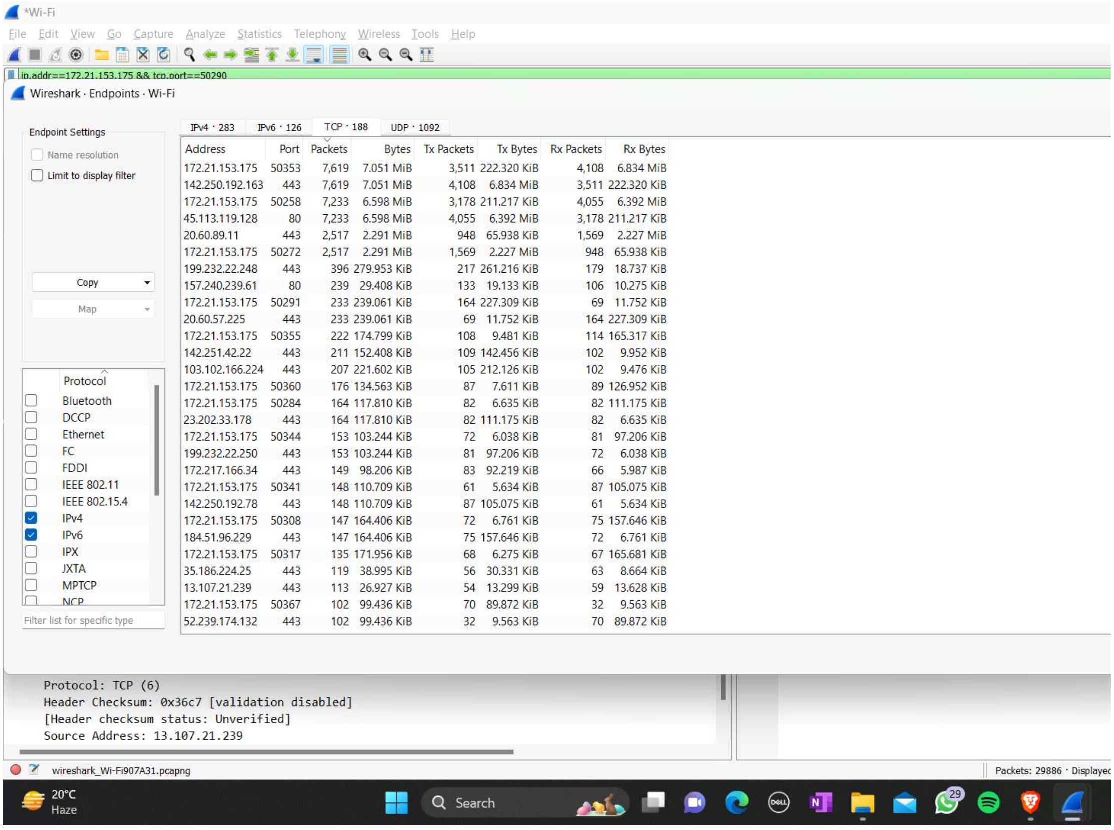
Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
00:0b:86:b2:ab:86	01:80:c2:00:00:00	43	2.855 KiB	43	2.855 KiB	0	0 bytes	1.752082	131.9638	177 bytes	0 bytes
00:1e:64:f6:0d:3d	01:00:5e:7f:ff:fa	19	5.620 KiB	19	5.620 KiB	0	0 bytes	3.698580	130.0035	354 bytes	0 bytes
00:1e:64:f6:0d:3d	ff:ff:ff:ff:ff:ff	3	180 bytes	3	180 bytes	0	0 bytes	64.790020	1.6630	865 bytes	0 bytes
00:45:e2:c9:35:c1	01:00:5e:7f:ff:fa	6	1.271 KiB	6	1.271 KiB	0	0 bytes	41.610405	19.8254	525 bytes	0 bytes
00:45:e2:c9:35:c1	ff:ff:ff:ff:ff:ff	3	180 bytes	3	180 bytes	0	0 bytes	110.199611	1.6011	899 bytes	0 bytes
00:9d:6b:04:a9:8c	01:00:5e:00:00:fb	7	945 bytes	7	945 bytes	0	0 bytes	36.676916	91.8914	82 bytes	0 bytes
00:9d:6b:04:a9:8c	01:00:5e:7f:ff:fa	41	6.827 KiB	41	6.827 KiB	0	0 bytes	1.062262	129.5556	431 bytes	0 bytes
00:db:df:3d:cd:2c	01:00:5e:7f:ff:fa	2	434 bytes	2	434 bytes	0	0 bytes	42.490388	3.0016	1.129 KiB	0 bytes
02:90:82:84:5f:3a	01:00:5e:00:00:fb	2	170 bytes	2	170 bytes	0	0 bytes	65.962203	1.0106	1.313 KiB	0 bytes
02:90:82:84:5f:3a	33:33:00:00:00:fb	2	210 bytes	2	210 bytes	0	0 bytes	65.967119	1.0057	1.631 KiB	0 bytes
06:6b:15:46:26:23	01:00:5e:00:00:fb	6	618 bytes	6	618 bytes	0	0 bytes	16.542590	100.2590	49 bytes	0 bytes
06:6b:15:46:26:23	01:00:5e:7f:ff:fa	20	3.232 KiB	20	3.232 KiB	0	0 bytes	1.920690	129.5800	204 bytes	0 bytes
08:5b:d6:ed:3b:9f	01:00:5e:7f:ff:fa	4	868 bytes	4	868 bytes	0	0 bytes	107.918499	2.0771	3.265 KiB	0 bytes
08:5b:d6:ed:3b:9f	ff:ff:ff:ff:ff:ff	1	60 bytes	1	60 bytes	0	0 bytes	20.196704	0.0000		
08:78:08:e4:a4:76	33:33:ff:e4:a4:76	1	78 bytes	1	78 bytes	0	0 bytes	2.874151	0.0000		
08:78:08:e4:a4:76	ff:ff:ff:ff:ff:ff	1	68 bytes	1	68 bytes	0	0 bytes	2.732675	0.0000		
0a:4f:b9:1d:20:bd	33:33:00:00:00:02	1	70 bytes	1	70 bytes	0	0 bytes	12.201816	0.0000		
0a:4f:b9:1d:20:bd	ff:ff:ff:ff:ff:ff	2	120 bytes	2	120 bytes	0	0 bytes	18.110692	60.1310	15 bytes	0 bytes
0a:8b:70:75:38:41	ff:ff:ff:ff:ff:ff	2	120 bytes	2	120 bytes	0	0 bytes	9.772522	3.0480	314 bytes	0 bytes
0a:a2:b7:5e:b4:ad	ff:ff:ff:ff:ff:ff	2	120 bytes	2	120 bytes	0	0 bytes	46.907497	60.1310	15 bytes	0 bytes
0c:e4:41:da:3f:56	01:00:5e:00:00:fb	1	920 bytes	1	920 bytes	0	0 bytes	87.262250	0.0000		
0c:e4:41:da:3f:56	01:00:5e:7f:ff:fa	3	651 bytes	3	651 bytes	0	0 bytes	60.263130	3.0243	1.682 KiB	0 bytes
0c:e4:41:da:3f:56	33:33:00:00:00:fb	1	940 bytes	1	940 bytes	0	0 bytes	87.263059	0.0000		
0c:e4:41:e1:a1:79	01:00:5e:00:00:fb	13	5.341 KiB	13	5.341 KiB	0	0 bytes	2.785965	120.0324	364 bytes	0 bytes
0c:e4:41:e1:a1:79	01:00:5e:7f:ff:fa	8	1.703 KiB	8	1.703 KiB	0	0 bytes	2.391511	123.0318	113 bytes	0 bytes
0c:e4:41:e1:a1:79	33:33:00:00:00:fb	14	5.982 KiB	14	5.982 KiB	0	0 bytes	2.787077	120.0320	408 bytes	0 bytes
0c:e4:41:e1:a1:79	ff:ff:ff:ff:ff:ff	1	250 bytes	1	250 bytes	0	0 bytes	77.431688	0.0000		
0e:2f:1e:99:5e:41	01:00:5e:00:00:fb	6	634 bytes	6	634 bytes	0	0 bytes	18.796910	100.3577	50 bytes	0 bytes

```
Protocol: TCP (6)
Header Checksum: 0x4e88 [validation disabled]
[Header checksum status: Unverified]
Source Address: 142.250.207.227
```

wireshark_Wi-Fi907A31.pcapng

Packets: 29886 · Displayed

- To view the data sent and received during the session, click on the "ASCII" or "Hex dump" buttons at the bottom of the window.
- Repeat the process for other packets with different TTL values to identify more TCP sessions.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr==172.21.153.175 && tcp.port==50290

Wireshark · Endpoints · Wi-Fi

Endpoint Settings

☐ Name resolution
☐ Limit to display filter

Protocol

☐ Bluetooth
☐ DCCP
☐ Ethernet
☐ FC
☐ FDDI
☐ IEEE 802.11
☐ IEEE 802.15.4
☑ IPv4
☑ IPv6
☐ IPX
☐ JXTA
☐ MPTCP
☐ NCP

Filter list for specific type

IPv4 · 283 | IPv6 · 126 | TCP · 188 | UDP · 1092

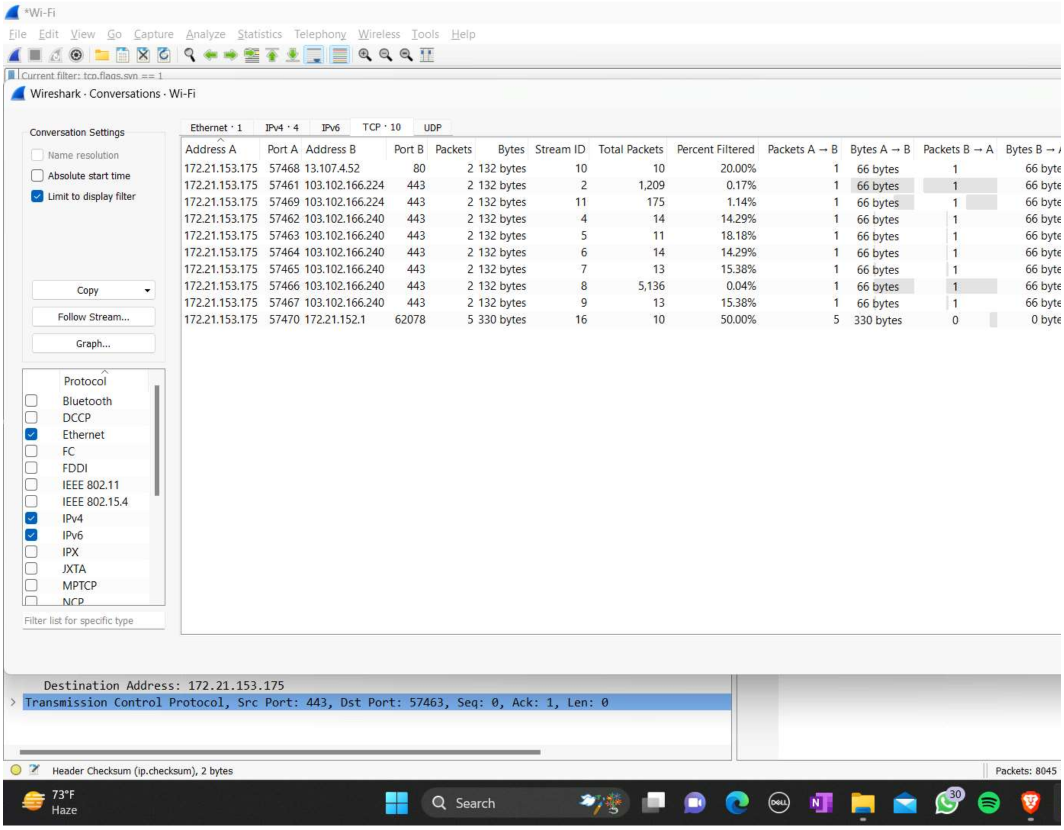
Address	Port	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
172.21.153.175	50353	7,619	7.051 MiB	3,511	222.320 KiB	4,108	6.834 MiB
142.250.192.163	443	7,619	7.051 MiB	4,108	6.834 MiB	3,511	222.320 KiB
172.21.153.175	50258	7,233	6.598 MiB	3,178	211.217 KiB	4,055	6.392 MiB
45.113.119.128	80	7,233	6.598 MiB	4,055	6.392 MiB	3,178	211.217 KiB
20.60.89.11	443	2,517	2.291 MiB	948	65.938 KiB	1,569	2.227 MiB
172.21.153.175	50272	2,517	2.291 MiB	1,569	2.227 MiB	948	65.938 KiB
199.232.22.248	443	396	279.953 KiB	217	261.216 KiB	179	18.737 KiB
157.240.239.61	80	239	29.408 KiB	133	19.133 KiB	106	10.275 KiB
172.21.153.175	50291	233	239.061 KiB	164	227.309 KiB	69	11.752 KiB
20.60.57.225	443	233	239.061 KiB	69	11.752 KiB	164	227.309 KiB
172.21.153.175	50355	222	174.799 KiB	108	9.481 KiB	114	165.317 KiB
142.251.42.22	443	211	152.408 KiB	109	142.456 KiB	102	9.952 KiB
103.102.166.224	443	207	221.602 KiB	105	212.126 KiB	102	9.476 KiB
172.21.153.175	50360	176	134.563 KiB	87	7.611 KiB	89	126.952 KiB
172.21.153.175	50284	164	117.810 KiB	82	6.635 KiB	82	111.175 KiB
23.202.33.178	443	164	117.810 KiB	82	111.175 KiB	82	6.635 KiB
172.21.153.175	50344	153	103.244 KiB	72	6.038 KiB	81	97.206 KiB
199.232.22.250	443	153	103.244 KiB	81	97.206 KiB	72	6.038 KiB
172.217.166.34	443	149	98.206 KiB	83	92.219 KiB	66	5.987 KiB
172.21.153.175	50341	148	110.709 KiB	61	5.634 KiB	87	105.075 KiB
142.250.192.78	443	148	110.709 KiB	87	105.075 KiB	61	5.634 KiB
172.21.153.175	50308	147	164.406 KiB	72	6.761 KiB	75	157.646 KiB
184.51.96.229	443	147	164.406 KiB	75	157.646 KiB	72	6.761 KiB
172.21.153.175	50317	135	171.956 KiB	68	6.275 KiB	67	165.681 KiB
35.186.224.25	443	119	38.995 KiB	56	30.331 KiB	63	8.664 KiB
13.107.21.239	443	113	26.927 KiB	54	13.299 KiB	59	13.628 KiB
172.21.153.175	50367	102	99.436 KiB	70	89.872 KiB	32	9.563 KiB
52.239.174.132	443	102	99.436 KiB	32	9.563 KiB	70	89.872 KiB

```
Protocol: TCP (6)
Header Checksum: 0x36c7 [validation disabled]
[Header checksum status: Unverified]
Source Address: 13.107.21.239
```

wireshark_Wi-Fi907A31.pcapng | Packets: 29886 · Displayed

It is important to note that the TTL field is not always reliable for identifying TCP sessions, as some network devices may modify the TTL value or use a different default value than the operating system. However, it can still be a useful tool in conjunction with other methods of session identification.

In addition, there are other fields in the IP header that can be used to identify TCP sessions, such as the source and destination IP addresses and port numbers. Wireshark also provides several other tools and filters that can aid in session identification and analysis, such as the "Follow TCP Stream" feature and the "Statistics" menu.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Current filter: tcp.flags.syn == 1
Wireshark · Conversations · Wi-Fi
Ethernet · 1
IPv4 · 4
IPv6
TCP · 10
UDP
Copy
Follow Stream...
Graph...
Filter list for specific type
73°F
Haze
Search

Conversation Settings

□ Name resolution
□ Absolute start time
☑ Limit to display filter

	Protocol
□	Bluetooth
□	DCCP
☑	Ethernet
□	FC
□	FDDI
□	IEEE 802.11
□	IEEE 802.15.4
☑	IPv4
☑	IPv6
□	IPX
□	JXTA
□	MPTCP
□	NCP

Address A	Port A	Address B	Port B	Packets	Bytes	Stream ID	Total Packets	Percent Filtered	Packets A → B	Bytes A → B	Packets B → A	Bytes B →
172.21.153.175	57468	13.107.4.52	80	2	132 bytes	10	10	20.00%	1	66 bytes	1	66 byte
172.21.153.175	57461	103.102.166.224	443	2	132 bytes	2	1,209	0.17%	1	66 bytes	1	66 byte
172.21.153.175	57469	103.102.166.224	443	2	132 bytes	11	175	1.14%	1	66 bytes	1	66 byte
172.21.153.175	57462	103.102.166.240	443	2	132 bytes	4	14	14.29%	1	66 bytes	1	66 byte
172.21.153.175	57463	103.102.166.240	443	2	132 bytes	5	11	18.18%	1	66 bytes	1	66 byte
172.21.153.175	57464	103.102.166.240	443	2	132 bytes	6	14	14.29%	1	66 bytes	1	66 byte
172.21.153.175	57465	103.102.166.240	443	2	132 bytes	7	13	15.38%	1	66 bytes	1	66 byte
172.21.153.175	57466	103.102.166.240	443	2	132 bytes	8	5,136	0.04%	1	66 bytes	1	66 byte
172.21.153.175	57467	103.102.166.240	443	2	132 bytes	9	13	15.38%	1	66 bytes	1	66 byte
172.21.153.175	57470	172.21.152.1	62078	5	330 bytes	16	10	50.00%	5	330 bytes	0	0 byte

```
    Destination Address: 172.21.153.175
>  Transmission Control Protocol, Src Port: 443, Dst Port: 57463, Seq: 0, Ack: 1, Len: 0
```

Header Checksum (ip.checksum), 2 bytes   Packets: 8045

Overall, the TTL field in the IP header is a valuable tool in identifying TCP sessions in Wireshark, but it should be used in combination with other methods and tools for more accurate and comprehensive analysis.

Finding Open Ports

1. Open Wireshark and load the captured network traffic file.

● Launch Wireshark on your computer and load the captured network traffic file that you want to analyze. You can do this by clicking on "File" > "Open" and selecting the file from your computer.

2. Go to the "Statistics" menu and select "Conversations."

● Once you have loaded the capture file, go to the "Statistics" menu at the top of the Wireshark window and select "Conversations" from the drop-down menu. This will bring up a new window that shows you the conversations that took place during the capture.

3. Select the "TCP" tab.

● In the "Conversations" window, you will see several tabs at the top. Select the "TCP" tab to view the TCP conversations that took place during the capture.

4. Sort the list based on the services used during these conversations.

● In the TCP tab, you will see a list of all the TCP conversations that took place during the capture. To sort the list based on the services used during these conversations, click on the "Service" column.

5. Look for the services that are using a high number of ports.

● Once you have sorted the list by the "Service" column, look for the services that are using a high number of ports. These are the services that are likely to have multiple open ports.

6. Select the service you want to investigate and click on the "Follow" button to view the details of the selected conversation.

● To investigate a specific service, select the row that corresponds to that service and click on the "Follow" button. This will bring up a new window that shows you the details of the selected conversation.

7. View the details of the conversation, including the ports that were used.

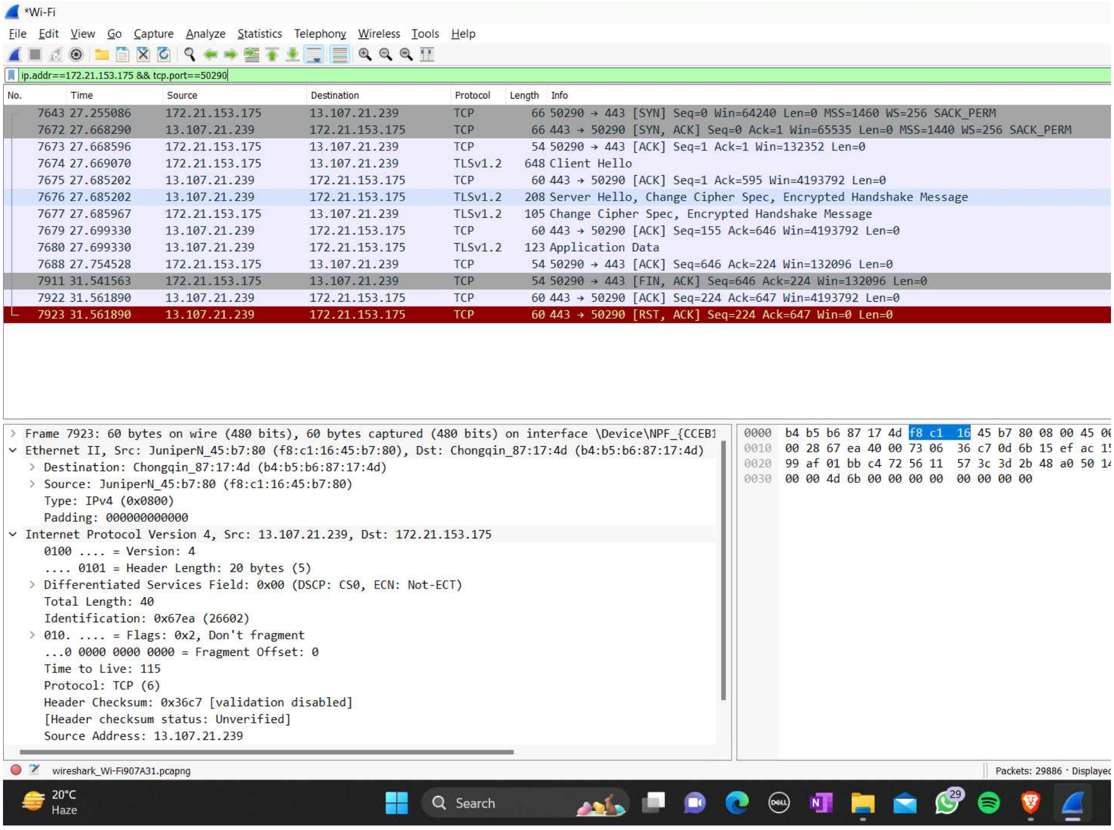
● In the "Follow TCP Stream" window, you will see the details of the conversation, including the ports that were used. You can view the source and destination IP addresses, the source and destination ports, and the data that was exchanged during the conversation.

8. Note down the ports that were used for the selected service.

● After viewing the details of the conversation, note down the ports that were used for the selected service. These are the open ports for that service.

9. Repeat the above steps for each service you want to investigate.

● To find open ports for other services, repeat the above steps for each service that you want to investigate.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr==172.21.153.175 && tcp.port==50290

```
No.   Time       Source          Destination     Protocol Length Info
 7643 27.255086  172.21.153.175  13.107.21.239   TCP       66 50290 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
 7672 27.668290  13.107.21.239   172.21.153.175  TCP       66 443 → 50290 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM
 7673 27.668596  172.21.153.175  13.107.21.239   TCP       54 50290 → 443 [ACK] Seq=1 Ack=1 Win=132352 Len=0
 7674 27.669070  172.21.153.175  13.107.21.239   TLSv1.2  648 Client Hello
 7675 27.685202  13.107.21.239   172.21.153.175  TCP       60 443 → 50290 [ACK] Seq=1 Ack=595 Win=4193792 Len=0
 7676 27.685202  13.107.21.239   172.21.153.175  TLSv1.2  208 Server Hello, Change Cipher Spec, Encrypted Handshake Message
 7677 27.685967  172.21.153.175  13.107.21.239   TLSv1.2  105 Change Cipher Spec, Encrypted Handshake Message
 7679 27.699330  13.107.21.239   172.21.153.175  TCP       60 443 → 50290 [ACK] Seq=155 Ack=646 Win=4193792 Len=0
 7680 27.699330  13.107.21.239   172.21.153.175  TLSv1.2  123 Application Data
 7688 27.754528  172.21.153.175  13.107.21.239   TCP       54 50290 → 443 [ACK] Seq=646 Ack=224 Win=132096 Len=0
 7911 31.541563  172.21.153.175  13.107.21.239   TCP       54 50290 → 443 [FIN, ACK] Seq=646 Ack=224 Win=132096 Len=0
 7922 31.561890  13.107.21.239   172.21.153.175  TCP       60 443 → 50290 [ACK] Seq=224 Ack=647 Win=4193792 Len=0
 7923 31.561890  13.107.21.239   172.21.153.175  TCP       60 443 → 50290 [RST, ACK] Seq=224 Ack=647 Win=0 Len=0
```

```
> Frame 7923: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{CCEB1
v Ethernet II, Src: JuniperN_45:b7:80 (f8:c1:16:45:b7:80), Dst: Chongqin_87:17:4d (b4:b5:b6:87:17:4d)
  > Destination: Chongqin_87:17:4d (b4:b5:b6:87:17:4d)
  > Source: JuniperN_45:b7:80 (f8:c1:16:45:b7:80)
    Type: IPv4 (0x0800)
    Padding: 000000000000
v Internet Protocol Version 4, Src: 13.107.21.239, Dst: 172.21.153.175
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 40
    Identification: 0x67ea (26602)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 115
    Protocol: TCP (6)
    Header Checksum: 0x36c7 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 13.107.21.239
```

```
0000  b4 b5 b6 87 17 4d f8 c1  16 45 b7 80 08 00 45 0
0010  00 28 67 ea 40 00 73 06  36 c7 0d 6b 15 ef ac 1
0020  99 af 01 bb c4 72 56 11  57 3c 3d 2b 48 a0 50 1
0030  00 00 4d 6b 00 00 00 00  00 00 00 00
```

wireshark_Wi-Fi907A31.pcapng   Packets: 29886 · Displayec

By following these steps, you can easily find open ports in Wireshark by going from "Statistics" to "Conversations" and analyzing the TCP conversations that took place during the captured network traffic. This can be useful for identifying potential security vulnerabilities or troubleshooting network issues.

GITHUB LINK

https://github.com/pavan984878/INT301