



Military Institute of Science and Technology

Department of CSE

CSE310 (Computer Network Sessional)

Introduction

Wireshark is a free open source network protocol analyzer. It is used for network troubleshooting and communication protocol analysis. Wireshark captures network packets in real time and display them in human-readable format. It provides many advanced features including live capture and offline analysis, three-pane packet browser, coloring rules for analysis. In this Lab class we will cover Wireshark installation, packet capturing, and protocol analysis.

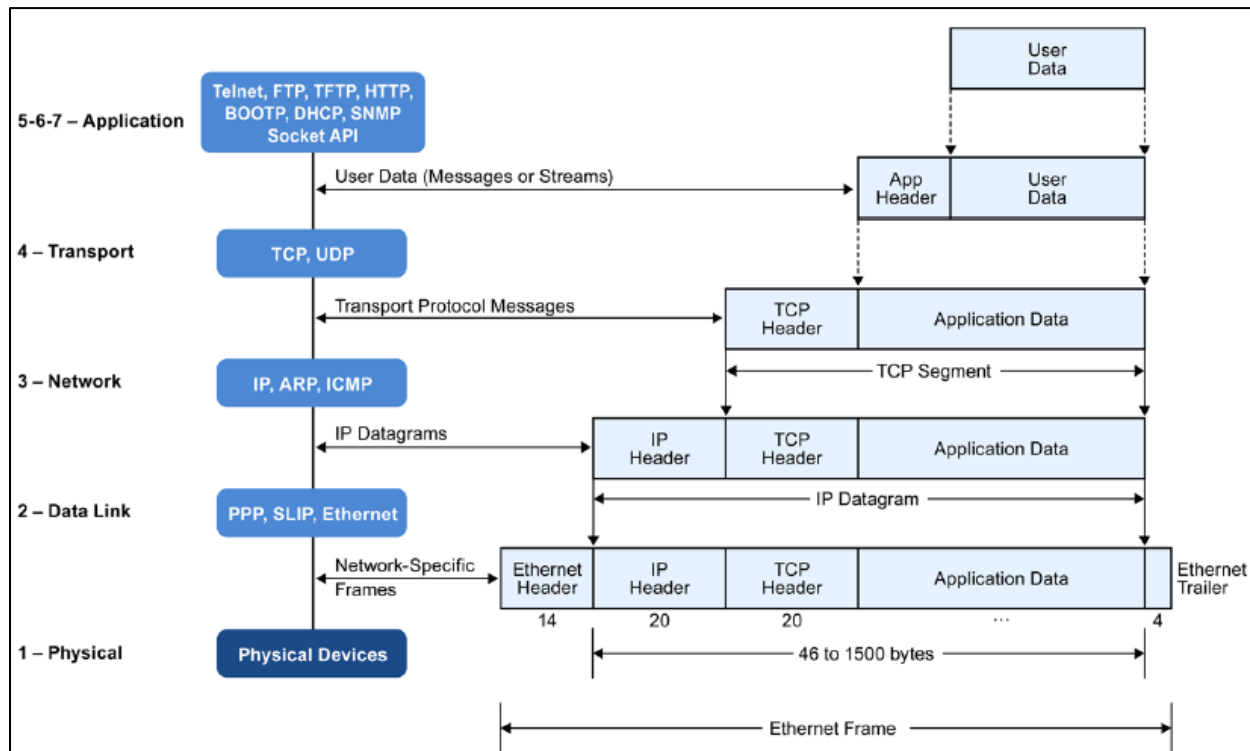


Figure-1: Encapsulation of Data in the TCP/IP Network

TCP/IP is the most commonly used network model for Internet services. Because its most important protocols, the Transmission Control Protocol (TCP) and the Internet Protocol (IP) were the first networking protocols defined in this standard, it is named as TCP/IP. However, it contains multiple layers including application layer, transport layer, network layer, and data link layer.



Introduction to Wireshark and Packet Sniffing

- *Application Layer:* The application layer includes the protocols used by most applications for providing user services. Examples of application layer protocols are Hypertext Transfer Protocol (HTTP), Secure Shell (SSH), File Transfer Protocol (FTP), and Simple Mail Transfer Protocol (SMTP).
- *Transport Layer:* The transport layer establishes process-to-process connectivity, and it provides end-to-end services that are independent of underlying user data. To implement the process-to-process communication, the protocol introduces a concept of port. The examples of transport layer protocols are Transport Control Protocol (TCP) and User Datagram Protocol (UDP). The TCP provides flow control, connection establishment, and reliable transmission of data, while the UDP is a connectionless transmission model.
- *Internet Layer:* The Internet layer is responsible for sending packets to across networks. It has two functions: 1) Host identification by using IP addressing system (IPv4 and IPv6); and 2) packets routing from source to destination. The examples of Internet layer protocols are Internet Protocol (IP), Internet Control Message Protocol (ICMP), and Address Resolution Protocol (ARP).
- *Link Layer:* The link layer defines the networking methods within the scope of the local network link. It is used to move the packets between two hosts on the same link. A common example of link layer protocols is Ethernet.

Packet Sniffer

Packet sniffer captures (“sniffs”) packets being sent/received from/by your computer; it will also typically store and/or display the contents of the various protocol fields in these captured packets. A packet sniffer itself is passive. It observes messages being sent and received by applications and protocols running on your computer, but never sends packets itself. **Figure-2** shows the structure of a packet sniffer.

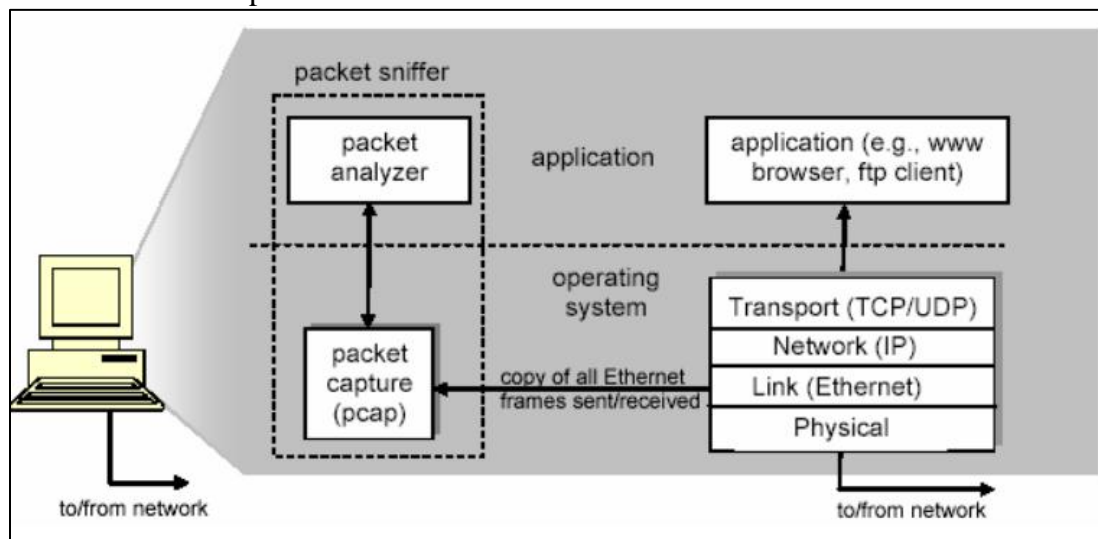


Figure-2: Packet Sniffer Structure



At the right of **Figure-2** are the protocols (in this case, Internet protocols) and applications (such as a web browser or ftp client) that normally run on your computer. The packet sniffer, shown within the dashed rectangle in **Figure-2** is an addition to the usual software in your computer, and consists of two parts. The packet capture library receives a copy of every link-layer frame that is sent from or received by your computer. Messages exchanged by higher layer protocols such as HTTP, FTP, TCP, UDP, DNS, or IP all are eventually encapsulated in link-layer frames that are transmitted over physical media such as an Ethernet cable. The assumed physical media is an Ethernet, and so all upper-layer protocols are eventually encapsulated within an Ethernet frame. Capturing all link-layer frames thus gives you access to all messages sent/received from/by all protocols and applications executing in your computer.

The second component of a packet sniffer is the packet analyzer, which displays the contents of all fields within a protocol message. In order to do so, the packet analyzer must “understand” the structure of all messages exchanged by protocols. For example, suppose we are interested in displaying the various fields in messages exchanged by the HTTP protocol in **Figure-2**. The packet analyzer understands the format of Ethernet frames, and so can identify the IP datagram within an Ethernet frame. It also understands the IP datagram format, so that it can extract the TCP segment within the IP datagram. Finally, it understands the TCP segment structure, so it can extract the HTTP message contained in the TCP segment. Finally, it understands the HTTP protocol and so, for example, knows that the first bytes of an HTTP message will contain the string “GET,” “POST,” or “HEAD”.

We will be using the Wireshark packet sniffer [<http://www.wireshark.org/>] for these labs, allowing us to display the contents of messages being sent/received from/by protocols at different levels of the protocol stack. (Technically speaking, Wireshark is a packet analyzer that uses a packet capture library in your computer). Wireshark is a free network protocol analyzer that runs on Windows, Linux/Unix, and Mac computers.

Why Wireshark

Wireshark is an open source packet sniffer and analysis tool. It captures packets on the local network from each connected devices to the network. This tool is used for monitoring traffic, troubleshooting in the network issues, inspecting individual packets. By this tool user can also detect suspicious activities in the network by analyzing packets.

Users of Wireshark

- Network administrator
- Cyber Security specialist
- Hacker



Introduction to Wireshark and Packet Sniffing

Getting Wireshark

The Kali Linux has Wireshark installed. You can just launch the Kali Linux VM and open Wireshark there. For Windows/macOS Wireshark can also be downloaded from here:

<https://www.wireshark.org/download.html>

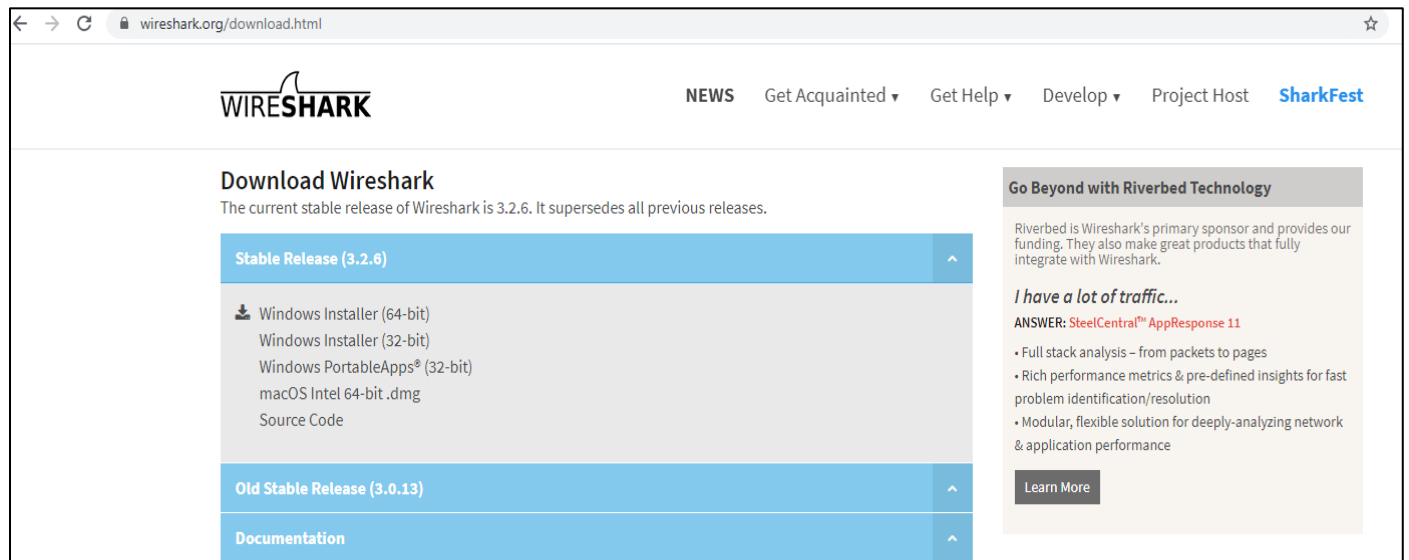


Figure-3: Download Page of Wireshark

Packet Capture Library Tools

- For Windows (Npcap or Winpcap)
- For Linux (Libpcap)
- For Wireless Network (Aircap)

Note: After installation remember to start Wireshark as Administrator. Otherwise the AirPCAP adapters are not found in the interface list.

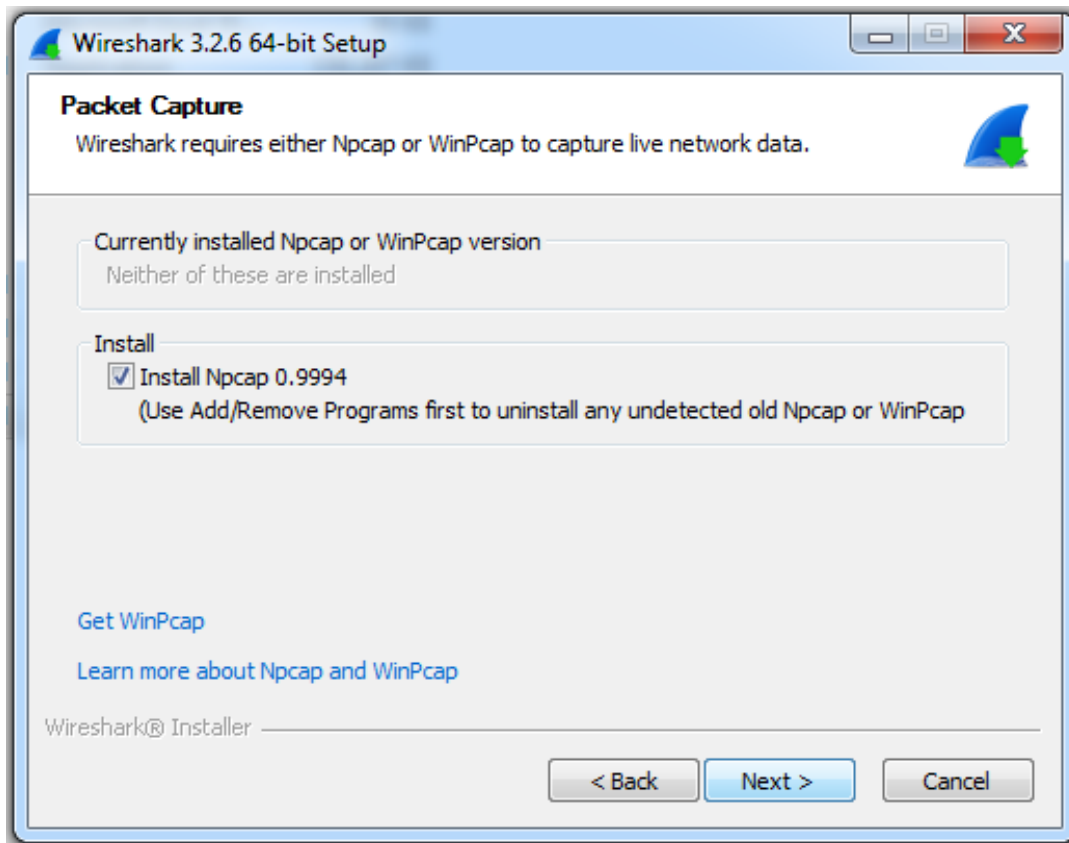
Types of Traffics Wireshark Captures

- Ethernet (LAN)
- Wireless (WLAN)
- Bluetooth
- USB



Windows Packet Capture (Npcap or Winpcap)

Npcap and WinPcap are Windows versions of the libpcap library. One of them must be installed in order to capture live network traffic on Windows. The Wireshark installer from 3.0 onwards includes Npcap, where versions before include WinPcap. Even with the older Wireshark versions Npcap might work better for you, especially if you run Windows 10.





Starting Wireshark

When we run the Wireshark program, the Wireshark graphic user interface will be shown as **Figure-4**. Currently, the program is not capturing the packets.

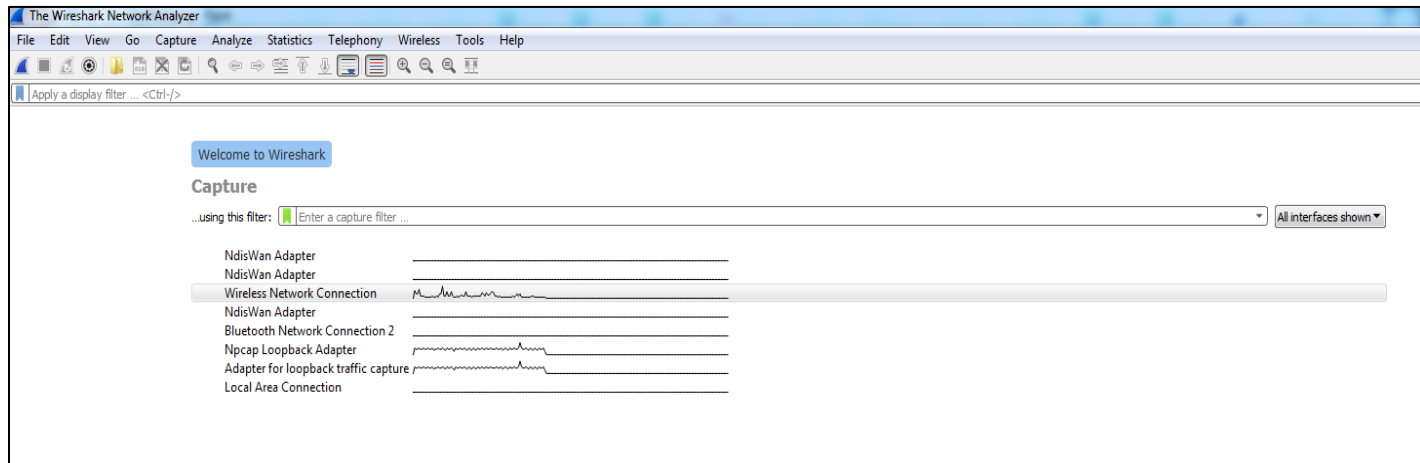


Figure-4: Initial Graphic User Interface of Wireshark

Capturing Packets

After downloading and installing Wireshark, we can launch it and click the name of an interface under Interface List to start capturing packets on that interface. For example, if we want to capture traffic on the Ethernet network, click Ethernet interface. If we want to capture traffic on the wireless network, click wireless interface and so on.

Color Coding

We'll probably see packets highlighted in green, blue, and black. Wireshark uses colors to help us identify the types of traffic at a glance. By default, green is TCP traffic, dark blue is DNS traffic, light blue is UDP traffic, and black identifies TCP packets with problems — for example, they could have been delivered out-of-order.

We need to choose an interface. If we are running the Wireshark on our laptop, we need to select WiFi interface. If we are at a desktop, we need to select the Ethernet interface being used. Note that there could be multiple interfaces. In general, we can select any interface but that does not mean that traffic will flow through that interface. The network interfaces (i.e., the physical connections) that our computer has to the network are shown. After we select the interface, we can click start to capture the packets as shown in **Figure-6**.



Introduction to Wireshark and Packet Sniffing

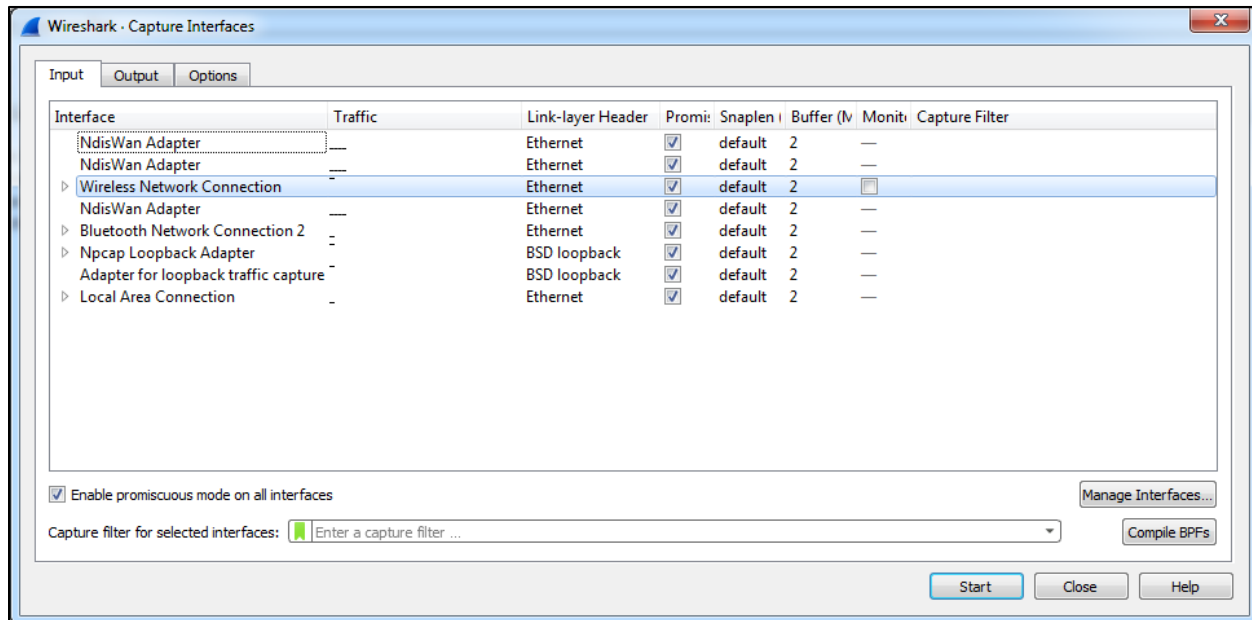


Figure-5: Capture Interfaces in Wireshark

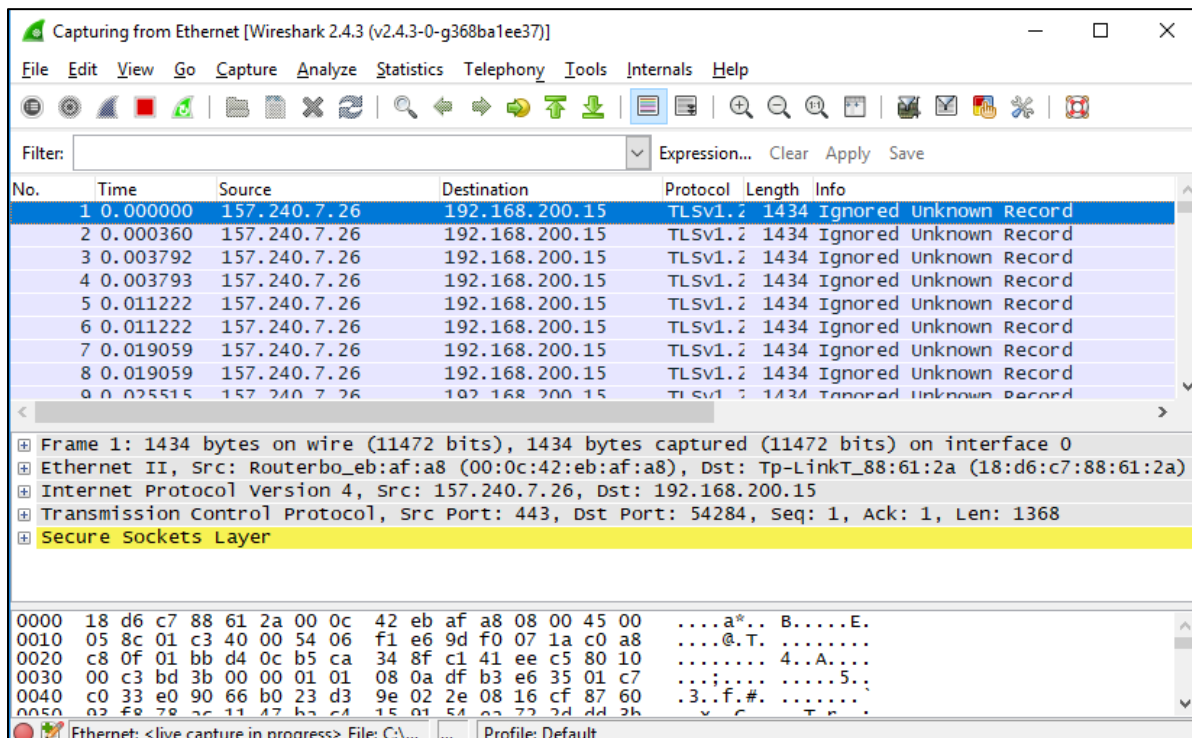


Figure-6: Capturing Packets in Wireshark



The Wireshark interface has five major components:

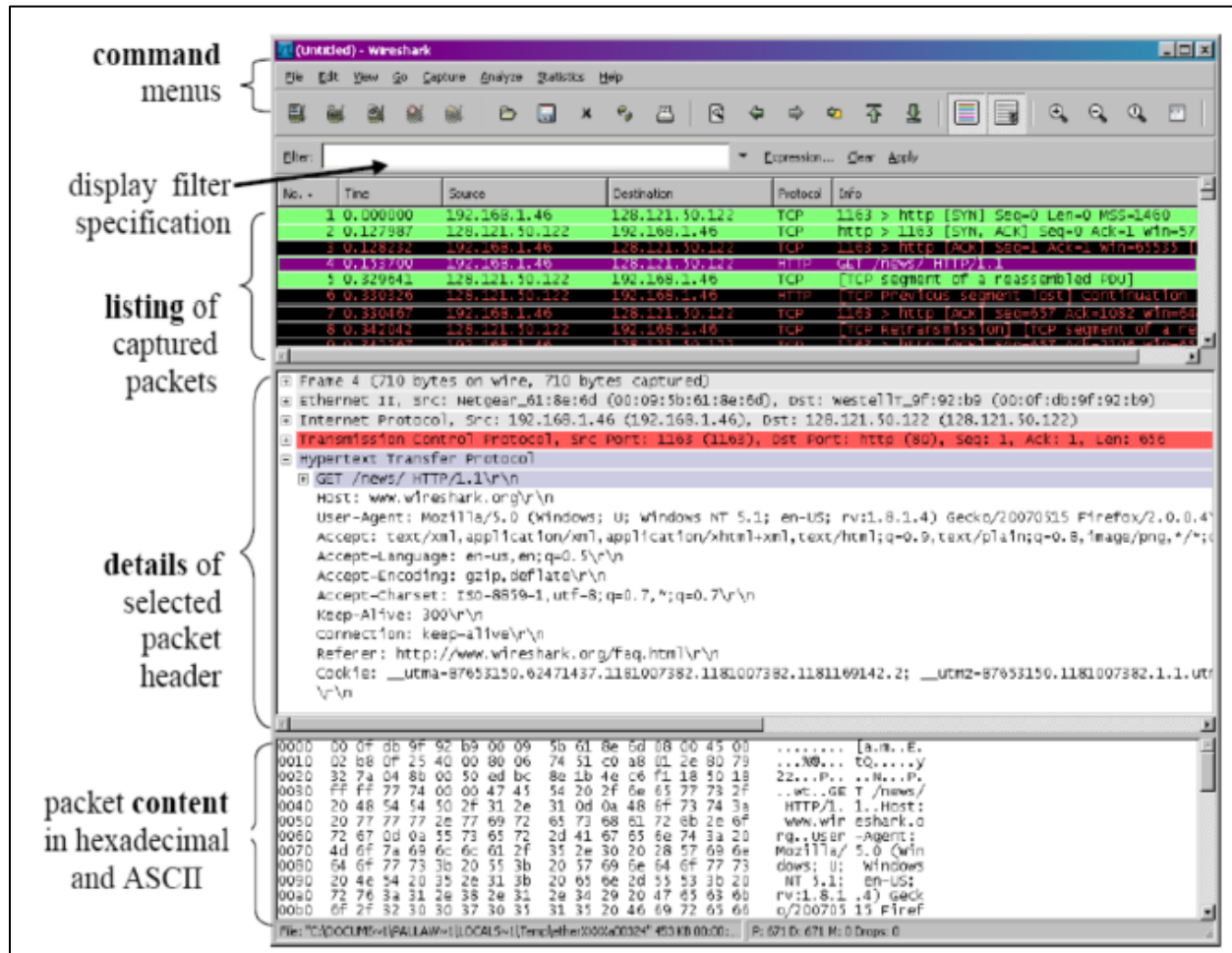


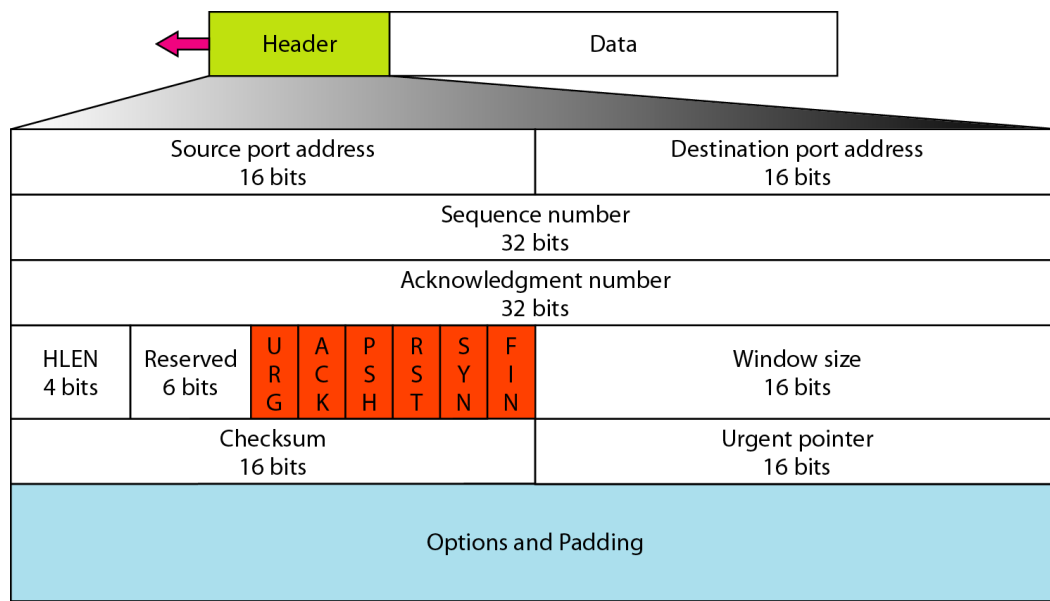
Figure-7: Wireshark Graphical User Interface on Microsoft Windows

The **command menus** are standard pulldown menus located at the top of the window. Of interest to us now is the File and Capture menus. The File menu allows you to save captured packet data or open a file containing previously captured packet data, and exit the Wireshark application. The Capture menu allows you to begin packet capture.

The **packet-listing window** displays a one-line summary for each packet captured, including the packet number (assigned by Wireshark; this is not a packet number contained in any protocol's header), the time at which the packet was captured, the packet's source and destination addresses, the protocol type, and protocol-specific information contained in the packet. The packet listing can be sorted according to any of these categories by clicking on a column name. The protocol type field lists the highest level protocol that sent or received this packet, i.e., the protocol that is the source or ultimate sink for this packet.



The **packet-header details window** provides details about the packet selected (highlighted) in the packet-listing window. (To select a packet in the packet-listing window, place the cursor over the packet's one-line summary in the packet-listing window and click with the left mouse button.). These details include information about the Ethernet frame and IP datagram that contains this packet. The amount of Ethernet and IP-layer detail displayed can be expanded or minimized by clicking on the right pointing or down-pointing arrowhead to the left of the Ethernet frame or IP datagram line in the packet details window. If the packet has been carried over TCP or UDP, TCP or UDP details will also be displayed, which can similarly be expanded or minimized. Finally, details about the highest-level protocol that sent or received this packet are also provided.



```

Transmission Control Protocol, Src Port: 443, Dst Port: 54284, Seq: 1, Ack: 1, Len: 1368
  Source Port: 443
  Destination Port: 54284
  [Stream index: 0]
  [TCP Segment Len: 1368]
  Sequence number: 1 (relative sequence number)
  [Next sequence number: 1369 (relative sequence number)]
  Acknowledgment number: 1 (relative ack number)
  1000 .... = Header Length: 32 bytes (8)
  + Flags: 0x010 (ACK)
    window size value: 195
    [Calculated window size: 195]
    [window size scaling factor: -1 (unknown)]
    Checksum: 0xbd3b [unverified]
    [Checksum status: unverified]
    Urgent pointer: 0
  + options: (12 bytes), No-operation (NOP), No-operation (NOP), Timestamps
  + [SEQ/ACK analysis]
    TCP payload (1368 bytes)
  
```

Figure-8: TCP Header



Introduction to Wireshark and Packet Sniffing

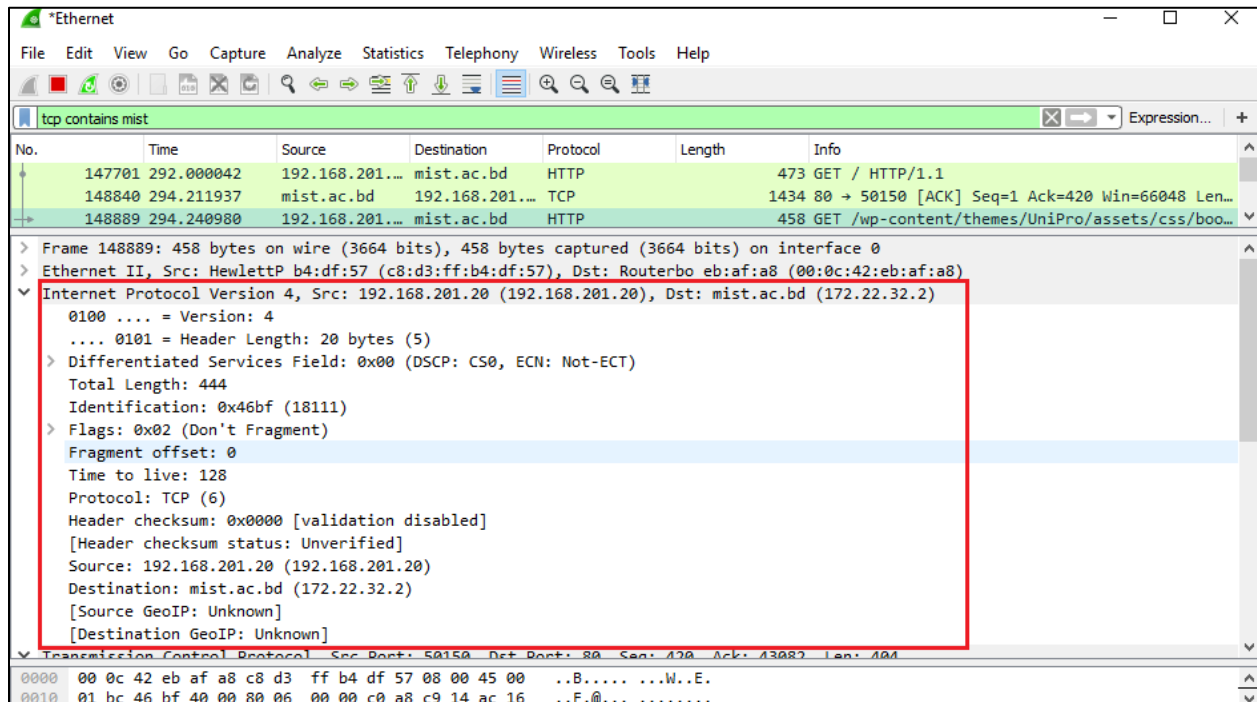
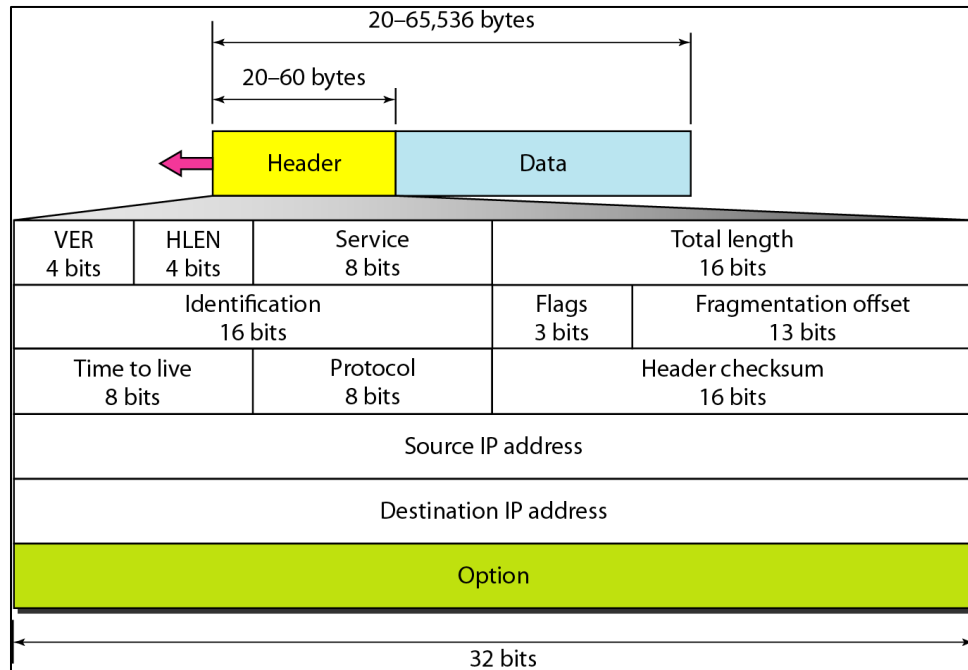


Figure-9: IPv4 Header



Introduction to Wireshark and Packet Sniffing

The **packet-contents window** displays the entire contents of the captured frame, in both ASCII and hexadecimal format.

Towards the top of the Wireshark graphical user interface, is the **packet display filter field**, into which a protocol name or other information can be entered in order to filter the information displayed in the packet-listing window (and hence the packet-header and packet-contents windows). In the example below, we'll use the packet-display filter field to have Wireshark hide (not display) packets except those that correspond to HTTP messages.

Filtering IP Address:

Wireshark 2.4.5 (v2.4.5-0-g153e867ef1) interface showing a packet capture on a wireless network connection. The filter field is set to `ip.addr==172.20.10.5`. The packet list shows several packets, including a SYN packet from 172.20.10.5 to 43.225.16.135. The packet details pane shows the structure of the selected packet: Ethernet II, Internet Protocol Version 4, and User Datagram Protocol. The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
8719	844.489626	172.20.10.5	122.180.236.126	UDP	62	52589 → 63006 Len=
8720	845.119912	172.20.10.5	43.225.16.135	TCP	66	3779 → 43611 [SYN]
8721	845.539640	172.20.10.5	59.152.105.42	UDP	62	52589 → 64116 Len=
8722	845.575320	43.225.16.135	172.20.10.5	TCP	58	43611 → 3779 [SYN]
8723	845.575445	172.20.10.5	43.225.16.135	TCP	54	3779 → 43611 [ACK]
8724	845.575546	172.20.10.5	43.225.16.135	TCP	54	[TCP window update]
8725	845.575676	172.20.10.5	43.225.16.135	BitTorr	122	Handshake

Frame 1: 1392 bytes on wire (11136 bits), 1392 bytes captured (11136 bits) on interface 802.11 Channel: Channel Offset: FCS Filter: All Frames None Wireless Settings... Decryption Keys...

Ethernet II, Src: ba:53:ac:22:86:64 (ba:53:ac:22:86:64), Dst: Azurewav_92:1d:c2 (1c:4 Internet Protocol Version 4, Src: 216.58.211.131, Dst: 172.20.10.5 User Datagram Protocol, Src Port: 443, Dst Port: 63887

0000 1c 4b d6 92 1d c2 ba 53 ac 22 86 64 08 00 45 00 .K....S..".d..E.
 0010 05 62 00 00 00 00 27 11 2c b4 d8 3a d3 83 ac 14 .b....'...:....
 0020 0a 05 01 bb f9 8f 05 4e a9 fc 04 9b f3 10 58 c4N.....X.
 0030 d3 31 18 27 34 2b a6 f4 fd 6d 24 08 c9 ce 5a e2 .1."4+..m\$....Z.
 0040 89 bc 3f 45 41 eb 0c b1 43 d3 4f 01 d3 e5 b0 40 ..?EA...C.O....@
 0050 2d 92 5f 5b 0e 00 8e bb 49 c3 14 4c b7 d7 bc 13 -.-[....I..L....
 0060 3d 3a 9c 7d 59 85 4a ad a2 3b 50 70 b8 fe 7b 21 =:.}Y.J. ;Pp..{!
 0070 d5 e8 ac 6a 3f e1 b4 48 ba 5e ed 5c 64 60 83 8d ...j?..H.^.\d`..

Wireless Network Connection: <live capture i... Packets: 8800 · Displayed: 8684 (98.7%) Profile: Default



Filtering TCP Protocol

Wireshark 2.4.5 (v2.4.5-0-g153e867ef1) interface showing a packet capture on a wireless network connection. The filter bar is set to 'tcp'. The packet list shows several TCP packets, with packet 13 (RST) highlighted in red. The packet details pane shows the structure of packet 13: Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol (RST). The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
9	0.139069	172.20.10.5	36.224.204.152	TCP	66	3246 → 17878 [SYN]
10	0.139182	172.20.10.5	160.202.145.35	TCP	66	3244 → 16883 [SYN]
11	0.139238	172.20.10.5	212.15.177.88	TCP	66	3245 → 35327 [SYN]
12	0.139284	172.20.10.5	39.53.249.206	TCP	66	3249 → 2939 [SYN]
13	0.302680	75.179.19.147	172.20.10.5	TCP	54	24874 → 3253 [RST]
27	2.122073	172.20.10.5	103.71.40.244	TCP	66	3251 → 61804 [SYN]
28	2.122072	172.20.10.5	39.45.205.222	TCP	66	3254 → 37588 [SYN]

Frame 9: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Ethernet II, Src: Azurewav_92:1d:c2 (1c:4b:d6:92:1d:c2), Dst: ba:53:ac:22:86:64 (ba:53:ac:22:86:64)
Internet Protocol Version 4, Src: 172.20.10.5, Dst: 36.224.204.152
Transmission Control Protocol, Src Port: 3246, Dst Port: 17878, Seq: 0, Len: 0

0000 ba 53 ac 22 86 64 1c 4b d6 92 1d c2 08 00 45 00 .S."d.KE.
0010 00 34 1c 31 40 00 80 06 37 01 ac 14 0a 05 24 e0 .4.1@... 7.....\$.
0020 cc 98 0c ae 45 d6 00 bb ba e1 00 00 00 00 80 02E.....
0030 20 00 99 63 00 00 02 04 05 b4 01 03 03 02 01 01 ...C.....
0040 04 02 ..

Wireshark 2.4.5 (v2.4.5-0-g153e867ef1) interface showing a packet capture on an Ethernet network connection. The filter bar is set to 'tcp.flags.syn==1'. The packet list shows several TCP packets, with packet 149110 (SYN) highlighted in red. The packet details pane shows the structure of packet 149110: Ethernet II, Destination: Routerbo_eb:af:a8 (00:0c:42:eb:af:a8), and Source: HewlettP_b4:df:57 (c8:d3:ff:b4:df:57). The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
148...	294.040794	api.apk.v-mate.mobi	192.168.202.213	TCP	66	80 → 55519 [SYN, ACK] Seq=0 Ack=1...
148...	294.242973	192.168.201.20	mist.ac.bd	TCP	66	50152 → 80 [SYN] Seq=0 Win=64240 ...
148...	294.243090	192.168.201.20	mist.ac.bd	TCP	66	50153 → 80 [SYN] Seq=0 Win=64240 ...
148...	294.243208	192.168.201.20	mist.ac.bd	TCP	66	50154 → 80 [SYN] Seq=0 Win=64240 ...
148...	294.247933	mist.ac.bd	192.168.201.20	TCP	66	80 → 50152 [SYN, ACK] Seq=0 Ack=1...
148...	294.247933	mist.ac.bd	192.168.201.20	TCP	66	80 → 50153 [SYN, ACK] Seq=0 Ack=1...
148...	294.247933	mist.ac.bd	192.168.201.20	TCP	66	80 → 50154 [SYN, ACK] Seq=0 Ack=1...
148...	294.250645	192.168.201.20	googleapis.l.google...	TCP	66	50155 → 80 [SYN] Seq=0 Win=64240 ...
148...	294.250739	192.168.201.20	googleapis.l.google...	TCP	66	50156 → 80 [SYN] Seq=0 Win=64240 ...
148...	294.299788	googleapis.l.google.com	192.168.201.20	TCP	66	80 → 50155 [SYN, ACK] Seq=0 Ack=1...

Frame 149110: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Ethernet II, Src: HewlettP_b4:df:57 (c8:d3:ff:b4:df:57), Dst: Routerbo_eb:af:a8 (00:0c:42:eb:af:a8)
Destination: Routerbo_eb:af:a8 (00:0c:42:eb:af:a8)
Source: HewlettP_b4:df:57 (c8:d3:ff:b4:df:57)

0000 00 0c 42 eb af a8 c8 d3 ff b4 df 57 08 00 45 00 ..B.....W..E.
0010 00 34 46 df 40 00 80 06 00 00 c0 a8 c9 14 ac 16 .4F.@... ..
0020 20 02 c3 ef 00 50 42 e8 bb 01 00 00 00 80 02PB.....
0030 fa f0 55 fc 00 00 02 04 05 b4 01 03 03 08 01 01 ..U.....
0040 04 02 ..



Capture ICMP Packets

Open the command prompt and ping any IP address (such as, google.com)

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Rania>ping google.com

Pinging google.com [74.125.130.100] with 32 bytes of data:
Reply from 74.125.130.100: bytes=32 time=157ms TTL=42
Reply from 74.125.130.100: bytes=32 time=143ms TTL=42
Reply from 74.125.130.100: bytes=32 time=217ms TTL=42
Reply from 74.125.130.100: bytes=32 time=155ms TTL=42

Ping statistics for 74.125.130.100:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 143ms, Maximum = 217ms, Average = 168ms

C:\Users\Rania>
```

Filter with ICMP packets and then observe the captured packets in the Wireshark. It will show the ICMP packets we are just ping.

Filter: icmp

No.	Time	Source	Destination	Protocol	Length	Info
9391	1270.80485	74.125.130.100	172.20.10.5	ICMP	74	Echo (ping) reply
9393	1271.64767	172.20.10.5	74.125.130.100	ICMP	74	Echo (ping) request
9394	1271.79084	74.125.130.100	172.20.10.5	ICMP	74	Echo (ping) reply
9395	1272.65390	172.20.10.5	74.125.130.100	ICMP	74	Echo (ping) request
9396	1272.87078	74.125.130.100	172.20.10.5	ICMP	74	Echo (ping) reply
9402	1273.65368	172.20.10.5	74.125.130.100	ICMP	74	Echo (ping) request
9403	1273.80921	74.125.130.100	172.20.10.5	ICMP	74	Echo (ping) reply

Frame 22: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0

- Ethernet II, Src: ba:53:ac:22:86:64 (ba:53:ac:22:86:64), Dst: Azurewav_92:1d:c2 (1c:4b:11:14:52:00)
- Internet Protocol Version 4, Src: 45.248.150.47, Dst: 172.20.10.5
- Internet Control Message Protocol

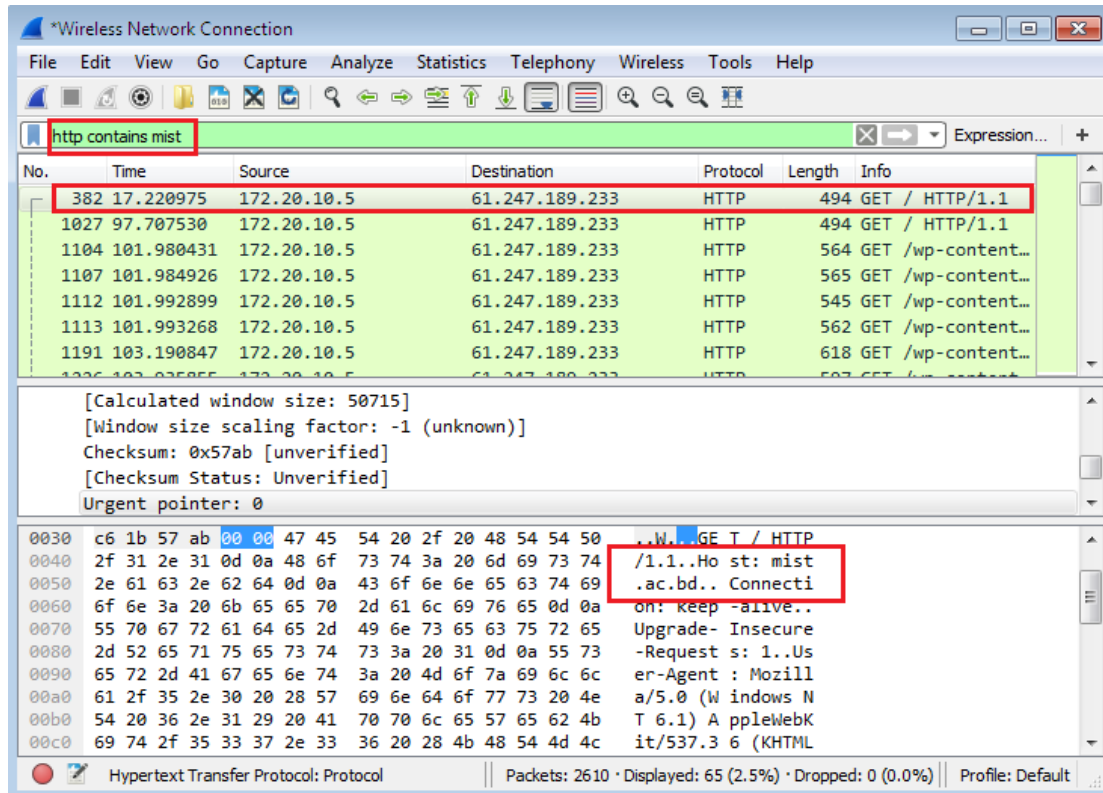
0000 1c 4b d6 92 1d c2 ba 53 ac 22 86 64 08 00 45 00 .K....S..".d..E.
 0010 00 38 af c3 00 00 36 01 5a c1 2d f8 96 2f ac 14 .8....6. Z.-.../.
 0020 0a 05 03 03 5f 59 00 00 00 00 45 00 00 30 1c 36Y.. ..E..0.6
 0030 00 00 76 11 ae 46 ac 14 0a 05 2d f8 96 2f cd 6d ..v..F..-.../.m
 0040 68 fc 00 1c 67 1d h...g.



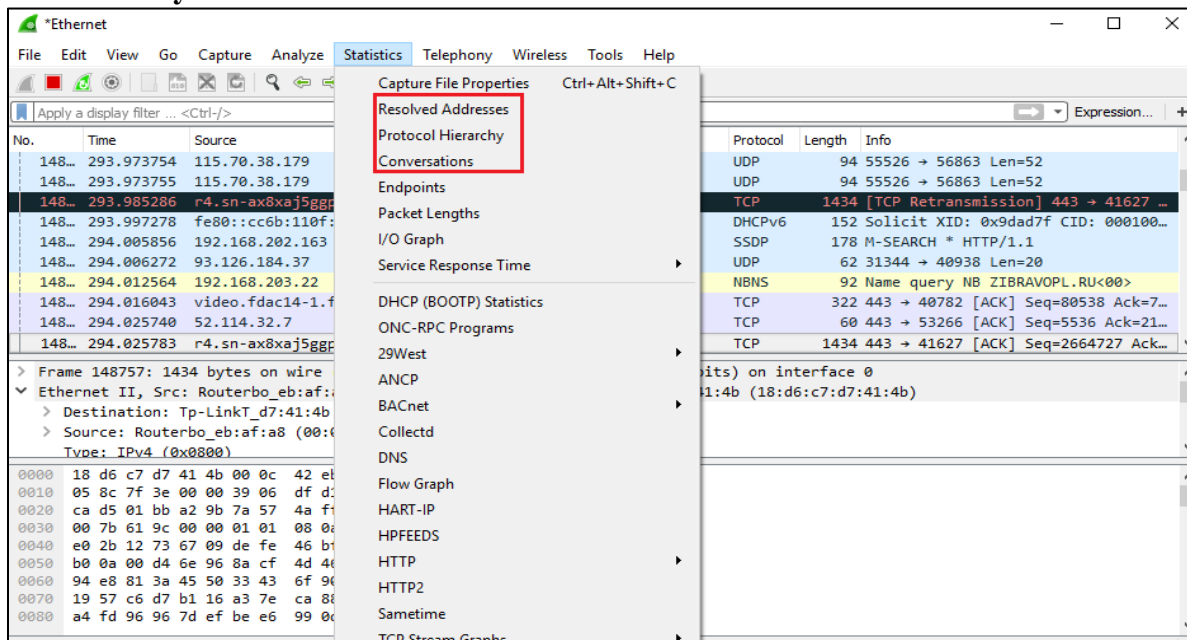
Introduction to Wireshark and Packet Sniffing

Capture HTTP packets

Open any website from any browser (such as, mist.ac.bd). Filter with **http contains mist**. Then observe the captured packets.



Statistics Analysis





Introduction to Wireshark and Packet Sniffing

Follow TCP stream

Filter with **tcp.port==80** (80 is the port number for http protocol)

The screenshot shows the Wireshark interface with the filter **tcp.port == 80** applied. The packet list displays several TCP packets from source 61.247.189.233 to destination 172.20.10.5. The packet details pane shows the selected packet (No. 262) with the following information:

- Source Port: 80
- Destination Port: 5411
- [Stream index: 12]
- [TCP Segment Len: 1380]
- Sequence number: 1 (relative sequence number)

The packet bytes pane shows the raw data of the packet, including the HTTP status line: **200 OK**.

Then follow TCP stream.

The screenshot shows the Wireshark interface with the filter **tcp.port == 80** applied. The packet list displays several TCP packets. The packet details pane shows the selected packet (No. 184) with the following information:

- Source Port: 80
- Destination Port: 5411
- [Stream index: 12]
- [TCP Segment Len: 1380]
- Sequence number: 1 (relative sequence number)

The packet bytes pane shows the raw data of the packet, including the HTTP status line: **200 OK**.

The 'Follow' menu is open, showing the following options:

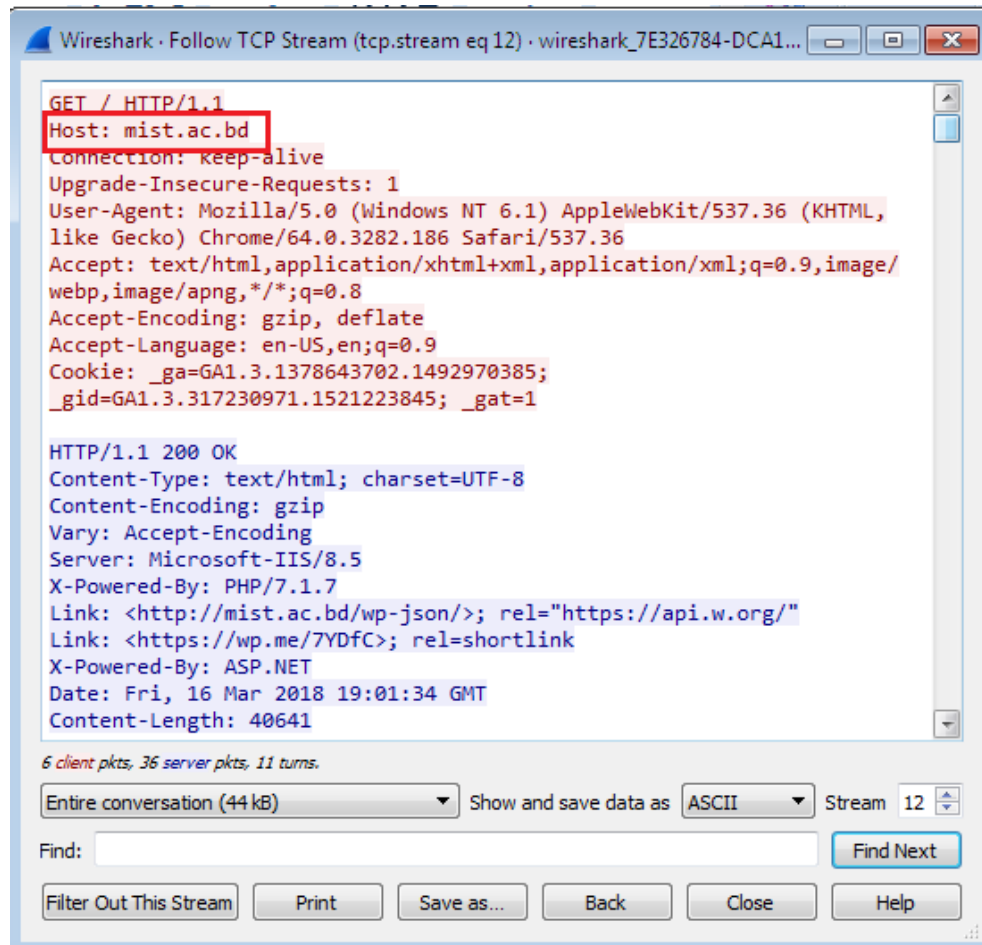
- Mark/Unmark Packet (Ctrl+M)
- Ignore/Unignore Packet (Ctrl+D)
- Set/Unset Time Reference (Ctrl+T)
- Time Shift... (Ctrl+Shift+T)
- Packet Comment... (Ctrl+Alt+C)
- Edit Resolved Name
- Apply as Filter
- Prepare a Filter
- Conversation Filter
- Colorize Conversation
- SCTP
- Follow
- Copy
- Protocol Preferences
- Decode As...
- Show Packet in New Window

The 'Follow' submenu is open, showing the following options:

- TCP Stream
- UDP Stream
- SSL Stream
- HTTP Stream



Red color is for client and Blue color is for server side. We can change the color code.



TCP Congestion Control Analysis

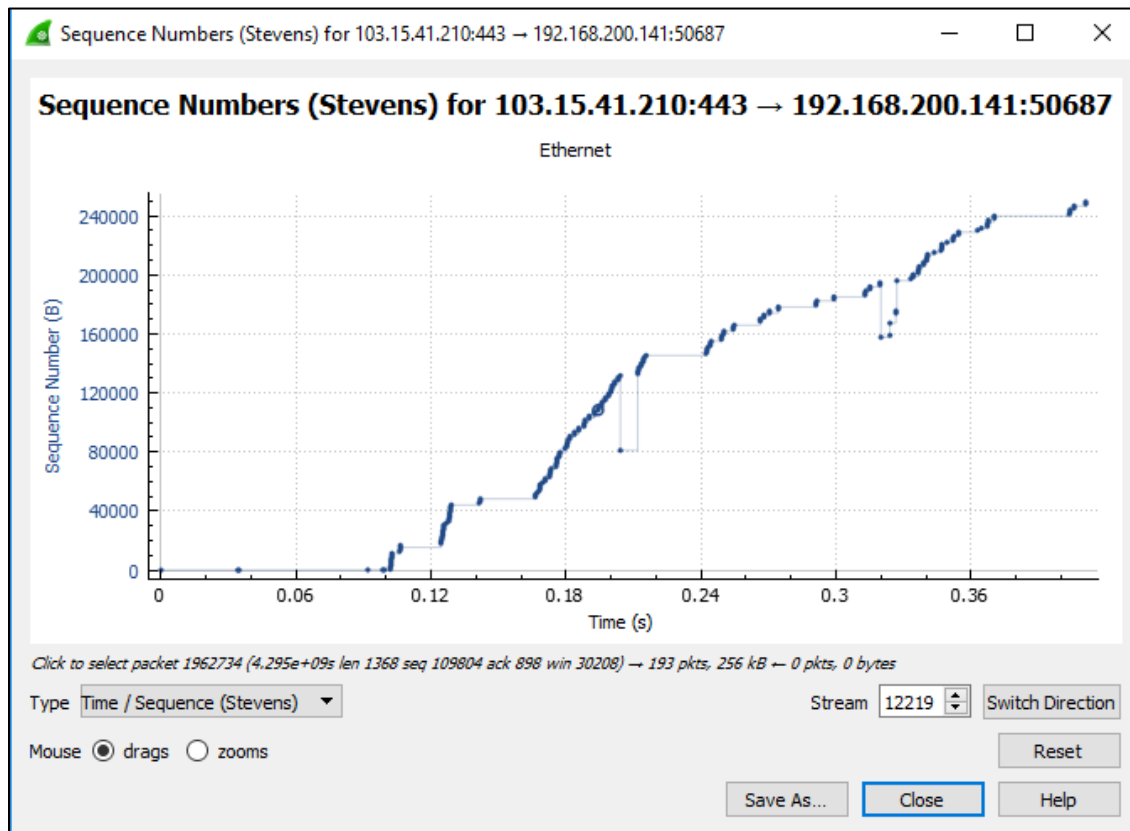
To avoid congestive collapse, TCP uses a multi-faceted congestion-control strategy. For each connection, TCP maintains a congestion window, limiting the total number of unacknowledged packets that may be in transit end-to-end. This is somewhat analogous to TCP's sliding window used for flow control. TCP uses a mechanism called slow start to increase the congestion window after a connection is initialized or after a timeout. It starts with a window of two times the maximum segment size (MSS). Although the initial rate is low, the rate of increase is very rapid; for every packet acknowledged, the congestion window increases by 1 MSS so that the congestion window effectively doubles for every round-trip time (RTT).

When the congestion window exceeds the slow-start threshold, $ssthresh$, the algorithm enters a new state, called congestion avoidance. In congestion avoidance state, as long as non-duplicate ACKs are received the congestion window is additively increased by one MSS every round-trip time.



Introduction to Wireshark and Packet Sniffing

The image shows the Wireshark interface with the 'Statistics' menu open. The 'Statistics' menu includes options like 'Capture File Properties', 'Resolved Addresses', 'Protocol Hierarchy', 'Conversations', 'Endpoints', 'Packet Lengths', 'I/O Graph', 'Service Response Time', 'DHCP (BOOTP) Statistics', 'ONC-RPC Programs', '29West', 'ANCP', 'BACnet', 'Collectd', 'DNS', 'Flow Graph', 'HART-IP', 'HPFEEDS', 'HTTP', 'HTTP2', 'Sametime', 'TCP Stream Graphs', 'UDP Multicast Streams', and 'IPv4 Statistics'. The 'TCP Stream Graphs' submenu is also visible, showing 'Time Sequence (Stevens)', 'Time Sequence (tcptrace)', and 'Throughput'. The main packet list shows a capture of traffic on the 'tcp' interface, with packets 1962784 through 1962791 listed. The packet details pane shows the selected packet (1962791) as a TCP segment with sequence number 1434 and destination port 50687. The packet bytes pane shows the raw data of the packet.

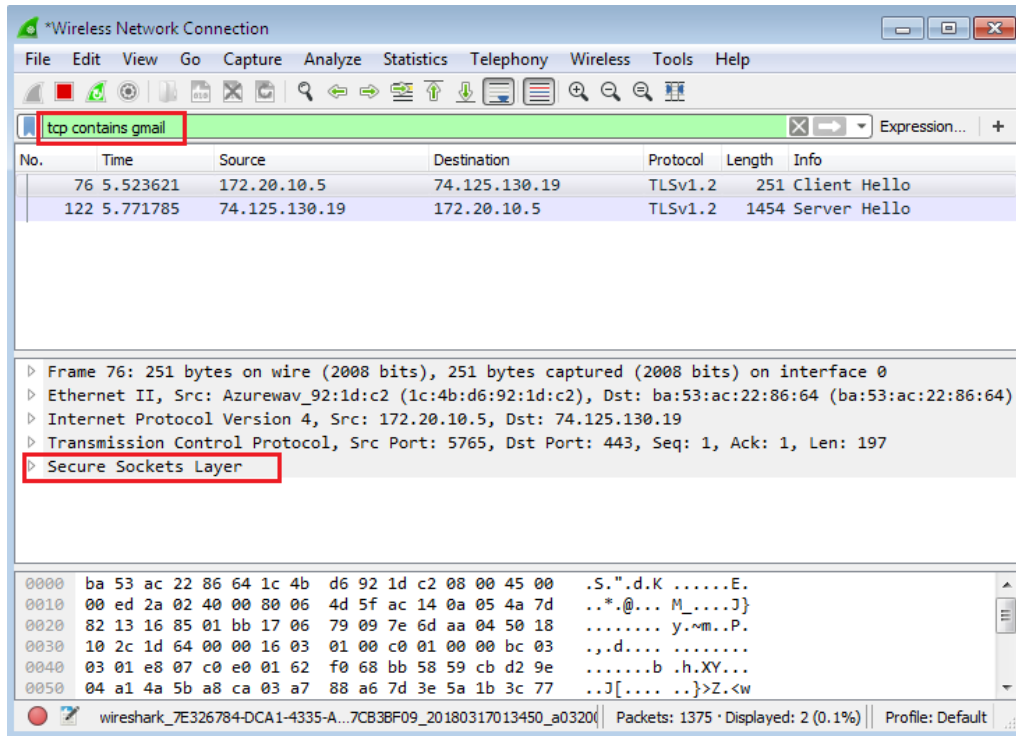




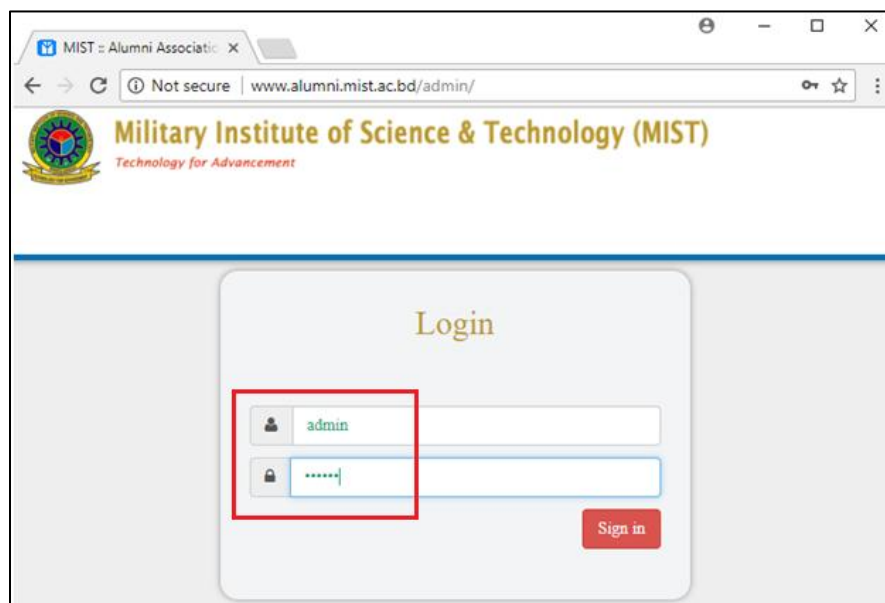
Introduction to Wireshark and Packet Sniffing

Sniffing Password

If we open **gmail.com** in a browser. Login and try to follow the TCP stream for sniffing username and password, we cannot. Because Gmail use SSL (Secured Socket Layer).



Now, open **www.alumni.mist.ac.bd/admin/** in a browser. Login with **username: admin** and **password: 123456**. There should be a message “Username and Password combination error!” because the password is incorrect.





Introduction to Wireshark and Packet Sniffing

Then, filter with **tcp contains mist** and follow the TCP stream.

Wireshark interface showing a packet list filtered by "tcp contains mist". The selected packet is 3209, an HTTP GET request. The packet details pane shows the structure of the HTTP request, including the status line "200 OK" and the response body.

No.	Time	Source	Destination	Protocol	Length	Info
3666	23.220641	216.12.210.66	192.168.201.20	HTTP	685	HTTP/1.1 404 Not Found (text/html)
319..	239.187787	192.168.201.20	216.12.210.66	HTTP	1038	POST /admin/ HTTP/1.1
320..	239.352513	216.12.210.66	192.168.201.20	TCP	1434	80 → 53640 [ACK] Seq=1381 Ack=985 Win...
320..	239.352625	216.12.210.66	192.168.201.20	TCP	1434	80 → 53640 [ACK] Seq=2761 Ack=985 Win...
320..	239.369910	192.168.201.20	216.12.210.66	HTTP	593	GET /admin/css/fonts/raleway-regular-...
320..	239.427293	216.12.210.66	192.168.201.20	HTTP	687	HTTP/1.1 404 Not Found (text/html)
320..	239.429066	192.168.201.20	216.12.210.66	HTTP	592	GET /admin/css/fonts/raleway-regular-...
320..	239.485493	216.12.210.66	192.168.201.20	HTTP	686	HTTP/1.1 404 Not Found (text/html)
320..	239.487739	192.168.201.20	216.12.210.66	HTTP	591	GET /admin/css/fonts/raleway-regular-...
320..	239.543606	216.12.210.66	192.168.201.20	HTTP	685	HTTP/1.1 404 Not Found (text/html)

> Frame 3209: 593 bytes on wire (4744 bits), 593 bytes captured (4744 bits) on interface 0
> Ethernet II, Src: HewlettP_b4:df:57 (c8:d3:ff:b4:df:57), Dst: RouterboEb:af:a8 (00:0c:42:eb:af:a8)
> Internet Protocol Version 4, Src: 192.168.201.20, Dst: 216.12.210.66
> Transmission Control Protocol, Src Port: 53640, Dst Port: 80, Seq: 985, Ack: 4268, Len: 539

0000 00 0c 42 eb af a8 c8 d3 ff b4 df 57 08 00 45 00 ..B.... ..W..E.
0010 02 43 49 fe 40 00 80 06 00 00 c0 a8 c9 14 d8 0c .CI.@... ..
0020 d2 42 d1 88 00 50 f8 a7 47 55 23 02 e0 ed 50 18 .B...P.. GU...P.
0030 01 02 36 42 00 00 47 45 54 20 2f 61 64 6d 69 6e ..6B...GE T /admin
0040 2f 63 73 73 2f 66 6f 6e 74 73 2f 72 61 6c 65 77 /css/fon ts/ralew
0050 61 79 2d 72 65 67 75 6c 61 72 2d 77 65 62 66 6f ay-regul ar-webfo
0060 6e 74 2e 77 6f 66 66 32 20 48 54 54 50 2f 31 2e nt.woff2 HTTP/1.
0070 31 0d 0a 48 6f 73 74 3a 20 77 77 77 2e 61 6c 75 1..Host: www.alu
0080 6d 6e 69 2e 6d 69 73 74 2e 61 63 2e 62 64 0d 0a mni.mist .ac.bd..
0090 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 6b 65 65 70 Connecti on: keep

Wireshark interface showing a packet list filtered by "tcp.stream eq 84". The selected packet is 7917, an HTTP POST request. The packet details pane shows the structure of the HTTP request, including the status line "200 OK" and the response body. A context menu is open over the packet list, showing options like "Mark/Unmark Packet", "Ignore/Unignore Packet", "Set/Unset Time Reference", "Time Shift...", "Packet Comment...", "Edit Resolved Name", "Apply as Filter", "Prepare a Filter", "Conversation Filter", "Colorize Conversation", "SCTP", "Follow", "Copy", and "Protocol Preferences".

No.	Time	Source	Destination	Protocol	Length	Info
7907	29.637925	192.168.201.20	216.12.210.66	TCP	66	50337 → 80 [SYN] Seq=0 Win=64240 Le...
7915	29.688413	216.12.210.66	192.168.201.20	TCP	66	80 → 50337 [SYN, ACK] Seq=0 Ack=1 W...
7916	29.688445	192.168.201.20	216.12.210.66	TCP	54	50337 → 80 [ACK] Seq=1 Ack=1 Win=66...
7917	29.688636	192.168.201.20	216.12.210.66	HTTP	1036	POST /admin/ HTTP/1.1
7924	29.739792	216.12.210.66	192.168.201.20	TCP	60	80 → 50337 [ACK] Seq=...
7925	29.750005	216.12.210.66	192.168.201.20	TCP	1434	80 → 50337 [ACK] Seq=...
7926	29.751283	216.12.210.66	192.168.201.20	TCP	1434	80 → 50337 [ACK] Seq=...
7927	29.751285	216.12.210.66	192.168.201.20	TCP	1434	80 → 50337 [ACK] Seq=...

> Frame 7917: 1036 bytes on wire (8288 bits), 1036 bytes captured (8288 bits) on interface 0
> Ethernet II, Src: HewlettP_b4:df:57 (c8:d3:ff:b4:df:57), Dst: RouterboEb:af:a8 (00:0c:42:eb:af:a8)
> Internet Protocol Version 4, Src: 192.168.201.20, Dst: 216.12.210.66
> Transmission Control Protocol, Src Port: 50337, Dst Port: 80, Seq: 1, Ack: 1, Len: 982

0000 00 0c 42 eb af a8 c8 d3 ff b4 df 57 08 00 45 00 ..B.... ..W..E.
0010 03 fe 75 5c 40 00 80 06 00 00 c0 a8 c9 14 d8 0c .u@... ..
0020 d2 42 c4 a1 00 50 e1 bf 5d 00 05 c9 23 e2 50 18 .B...P..]...#P.
0030 01 02 37 fd 00 00 50 4f 53 54 20 2f 61 64 6d 69 ..7...PO ST /admi
0040 6e 2f 20 48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 n/ HTTP/ 1.1..Hos
0050 74 3a 20 77 77 77 2e 61 6c 75 6d 6e 69 2e 6d 69 t: www.a lumni.mi
0060 73 74 2e 61 63 2e 62 64 0d 0a 43 6f 6e 6e 65 63 st.ac.bd ..Connec
0070 74 69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 65 tion: ke ep-alive
0080 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65 6e 67 74 68 ..Conten t-Length

Frame (frame), 1036 bytes | Packets: 349320

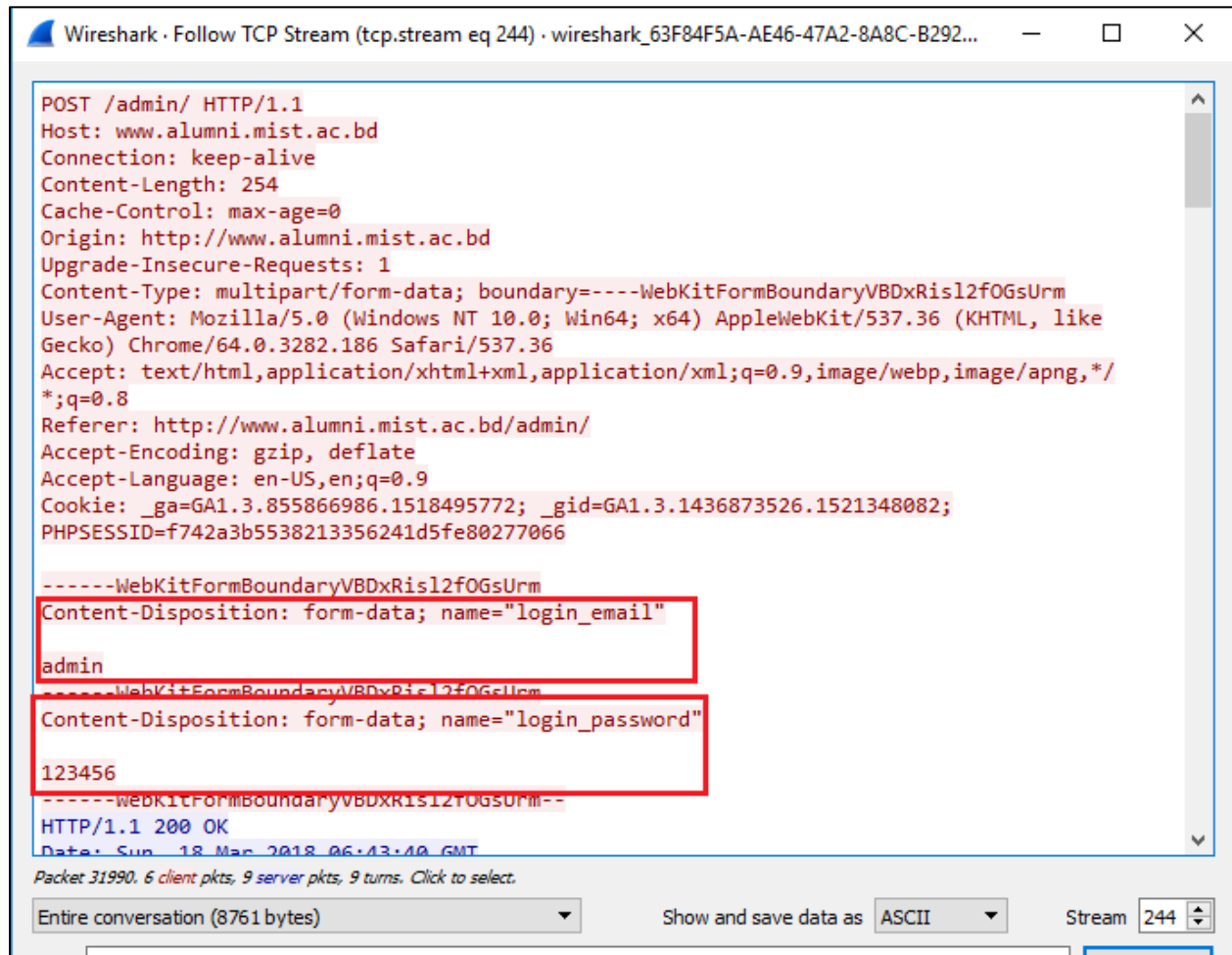
- Mark/Unmark Packet Ctrl+M
- Ignore/Unignore Packet Ctrl+D
- Set/Unset Time Reference Ctrl+T
- Time Shift... Ctrl+Shift+T
- Packet Comment... Ctrl+Alt+C
- Edit Resolved Name
- Apply as Filter
- Prepare a Filter
- Conversation Filter
- Colorize Conversation
- SCTP
- Follow
- Copy
- Protocol Preferences

TCP Stream
UDP Stream
SSL Stream
HTTP Stream



Introduction to Wireshark and Packet Sniffing

It will show the username and the password that we have used for login (though the password was wrong!).



The image shows a Wireshark packet capture window titled "Wireshark · Follow TCP Stream (tcp.stream eq 244) · wireshark_63F84F5A-AE46-47A2-8A8C-B292...". The main pane displays the details of an HTTP POST request to /admin/ on the host www.alumni.mist.ac.bd. The request includes various headers such as Connection: keep-alive, Content-Length: 254, Cache-Control: max-age=0, Origin: http://www.alumni.mist.ac.bd, Upgrade-Insecure-Requests: 1, Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryVBDxRis12fOGsUrm, User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36, Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8, Referer: http://www.alumni.mist.ac.bd/admin/, Accept-Encoding: gzip, deflate, Accept-Language: en-US,en;q=0.9, and cookies: _ga=GA1.3.855866986.1518495772; _gid=GA1.3.1436873526.1521348082; PHPSESSID=f742a3b5538213356241d5fe80277066. The body of the request is a multipart form-data with two parts: "login_email" with value "admin" and "login_password" with value "123456". Both parts are highlighted with red boxes. The response is HTTP/1.1 200 OK, dated Sun, 18 Mar 2018 06:43:40 GMT. The bottom status bar indicates "Packet 31990, 6 client pkts, 9 server pkts, 9 turns. Click to select." and the "Show and save data as" dropdown is set to "ASCII".

```
POST /admin/ HTTP/1.1
Host: www.alumni.mist.ac.bd
Connection: keep-alive
Content-Length: 254
Cache-Control: max-age=0
Origin: http://www.alumni.mist.ac.bd
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryVBDxRis12fOGsUrm
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/64.0.3282.186 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*
;q=0.8
Referer: http://www.alumni.mist.ac.bd/admin/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: _ga=GA1.3.855866986.1518495772; _gid=GA1.3.1436873526.1521348082;
PHPSESSID=f742a3b5538213356241d5fe80277066

-----WebKitFormBoundaryVBDxRis12fOGsUrm
Content-Disposition: form-data; name="login_email"

admin
-----WebKitFormBoundaryVBDxRis12fOGsUrm
Content-Disposition: form-data; name="login_password"

123456
-----WebKitFormBoundaryVBDxRis12fOGsUrm--
HTTP/1.1 200 OK
Date: Sun, 18 Mar 2018 06:43:40 GMT

Packet 31990, 6 client pkts, 9 server pkts, 9 turns. Click to select.
Entire conversation (8761 bytes) Show and save data as ASCII Stream 244
```