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.

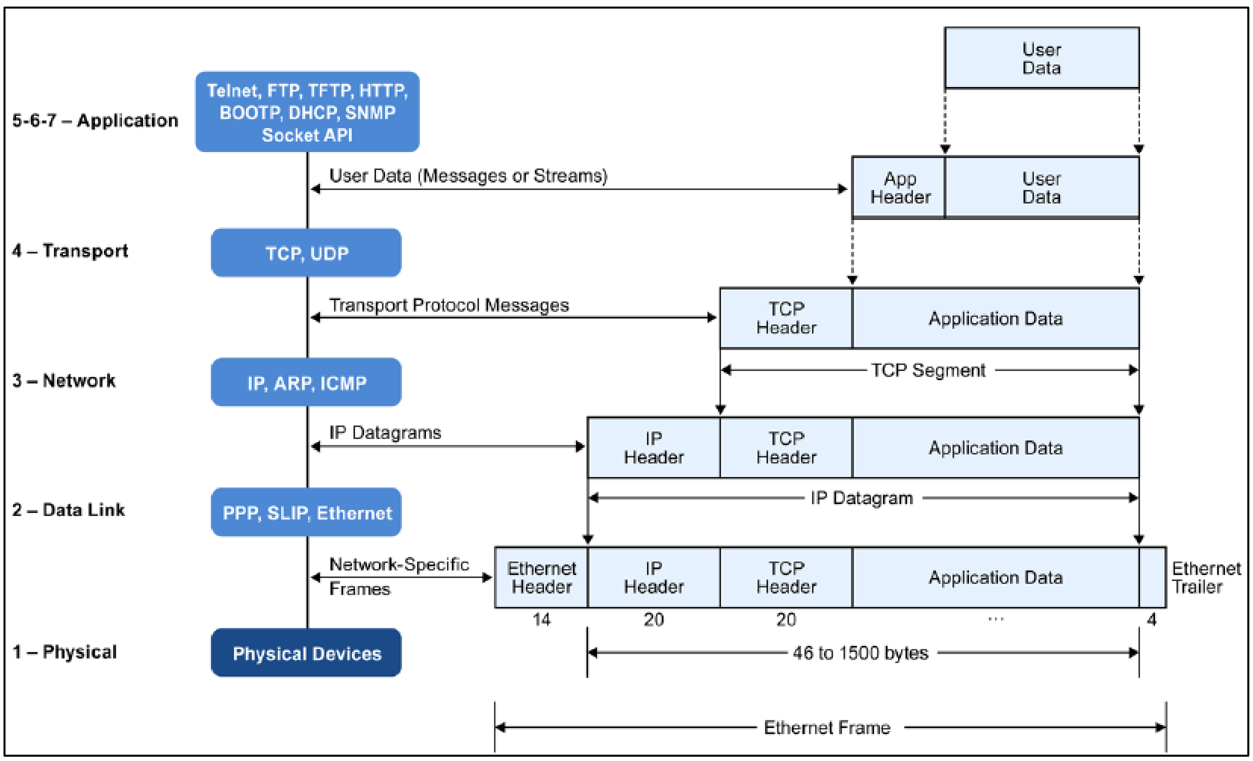


Figure-1: Encapsulation of Data in the TCP/IP Network

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.

- *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.

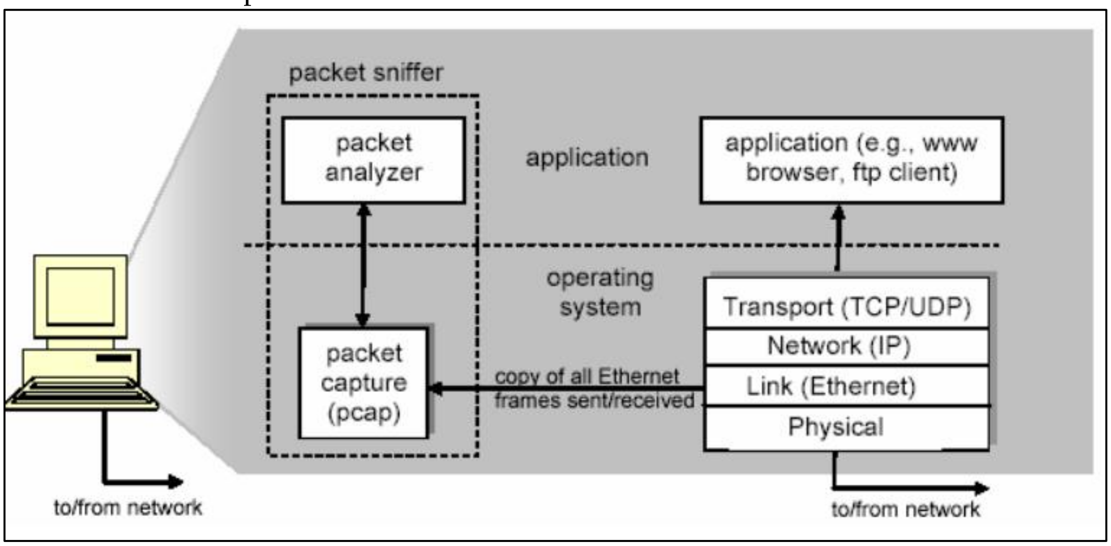


Figure-2: Packet Sniffer Structure

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

Getting Wireshark

The Kai 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

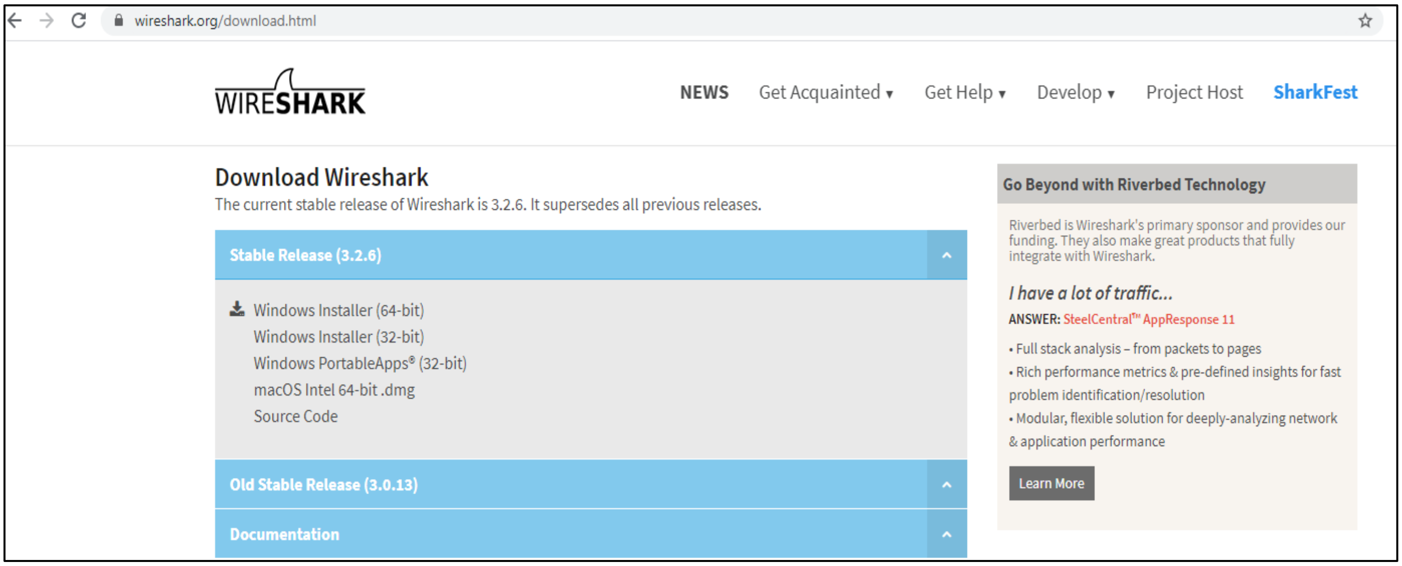


Figure-3: Download Page of Wireshark

Packet Capture Library Tools

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

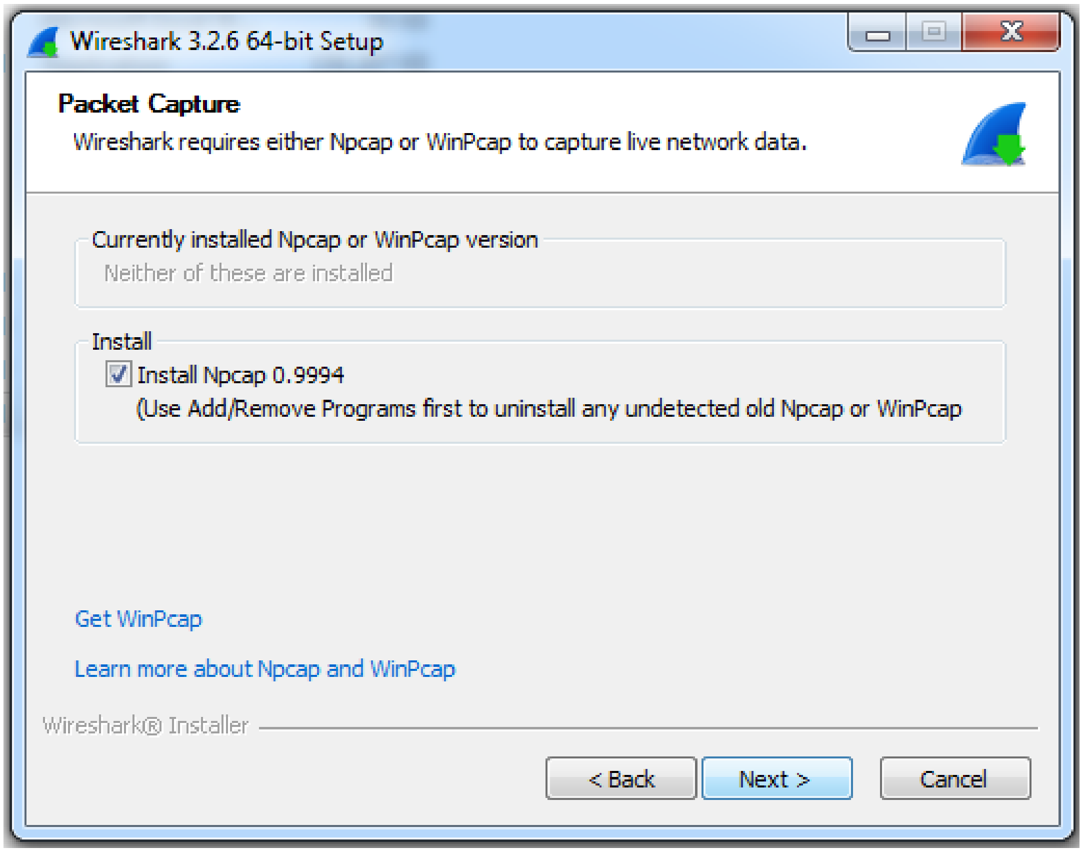
Note: After installation remember to start Wireshark as Adminstrator. Otherwise the AirPCAP adapters are not found in the interface list.

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.

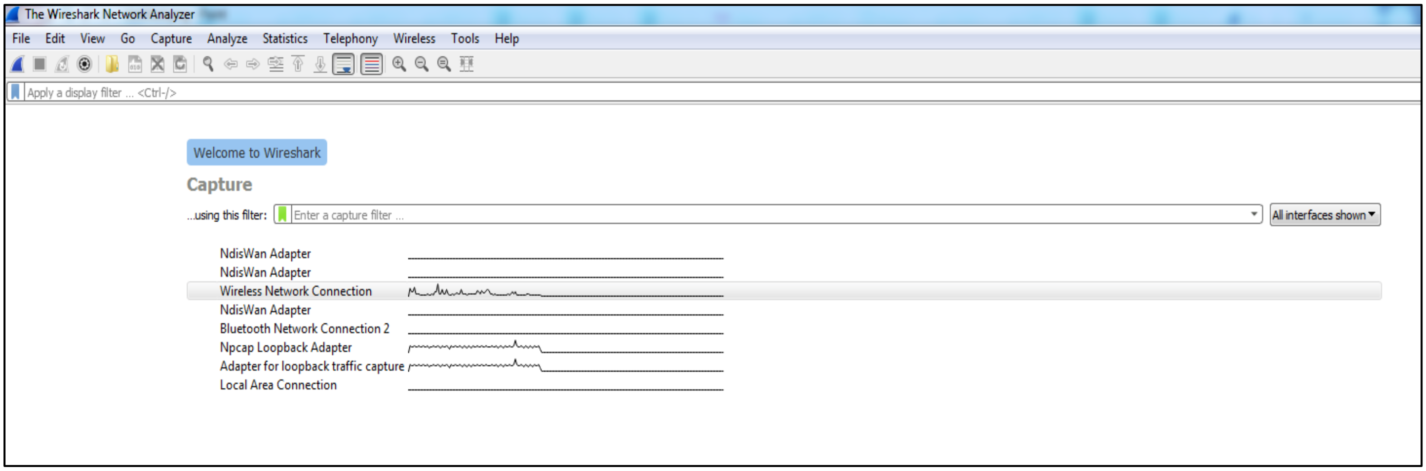


Figure-4: Initial Graphic User Interface of Wireshark

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**.

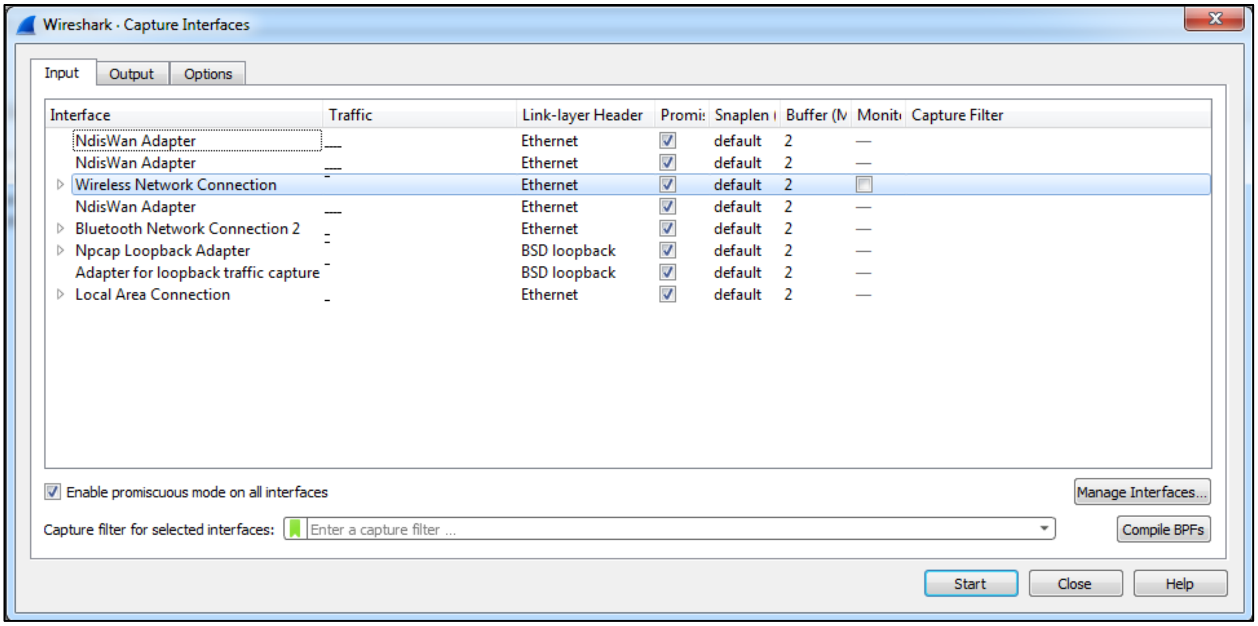


Figure-5: Capture Interfaces in Wireshark

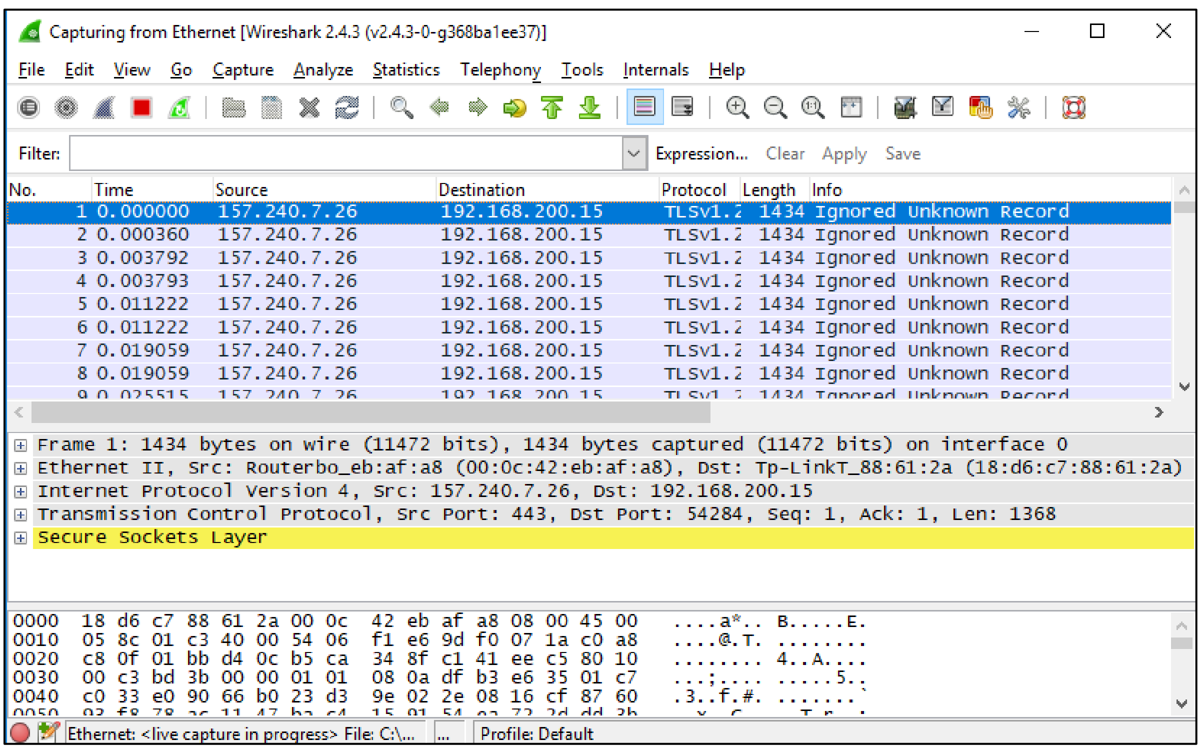


Figure-6: Capturing Packets in Wireshark

The Wireshark interface has five major components:

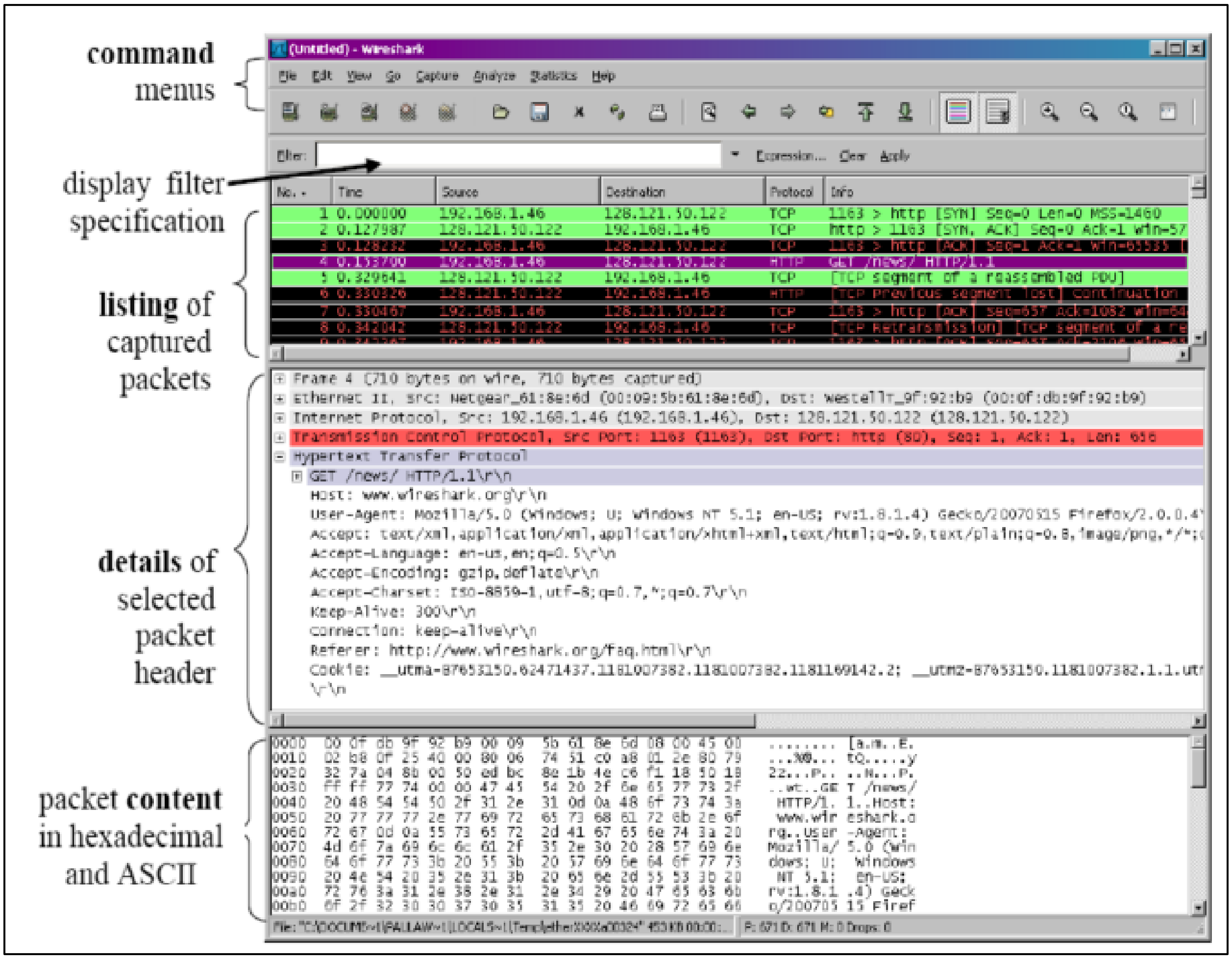
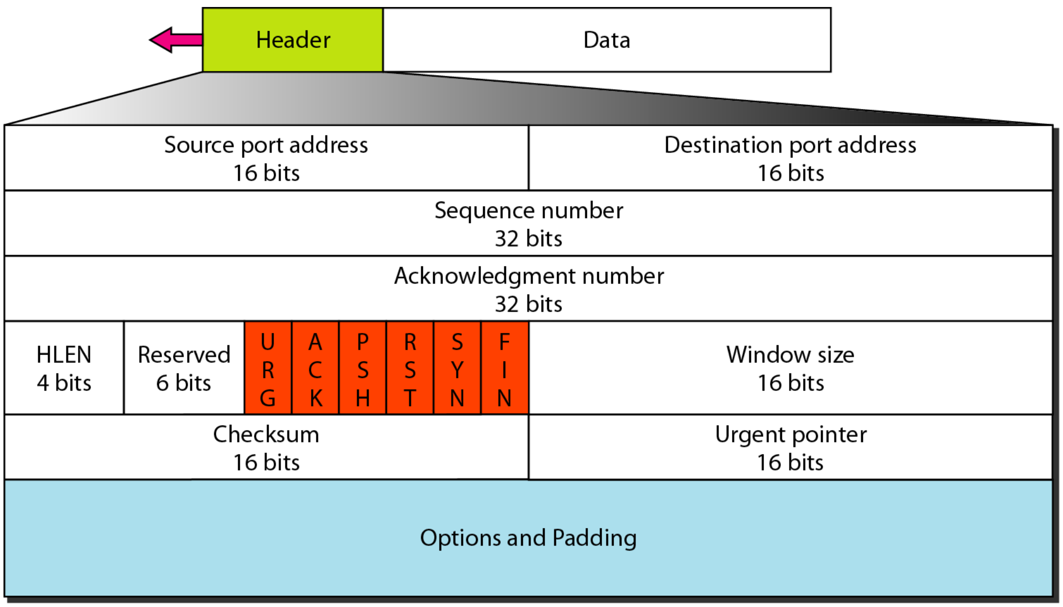


Figure-7: Wireshark Graphical User Interface on Microsoft Windows

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.

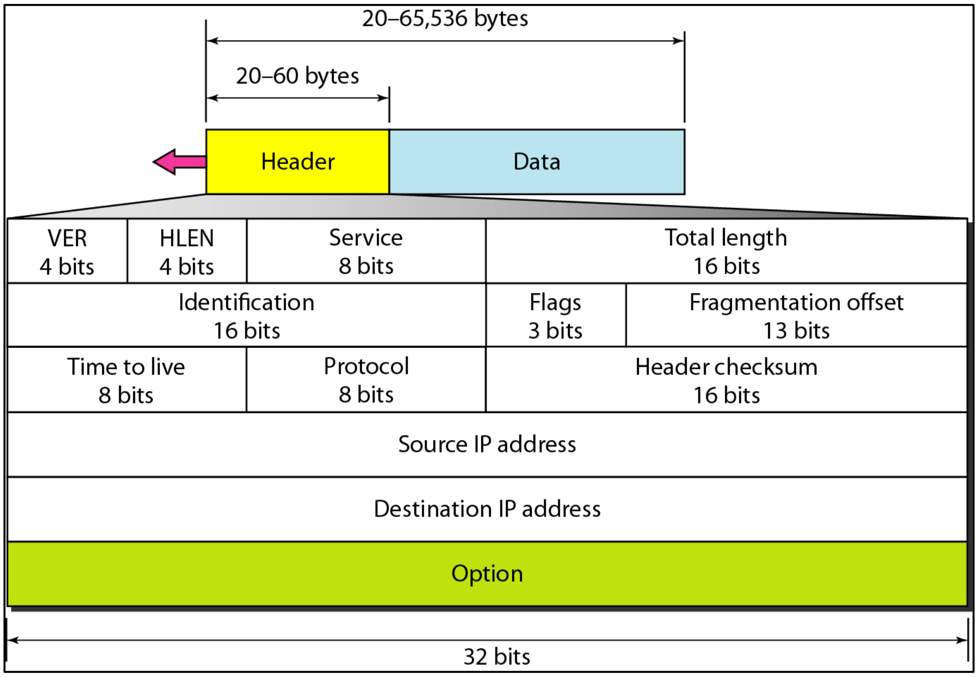
The **packet-listing window** displays a one-line summary for each packet captured, including the packet number (assigned by Wireshark; this is not a packet number contained in any protocol's header), the time at which the packet was captured, the packet's source and destination addresses, the protocol 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



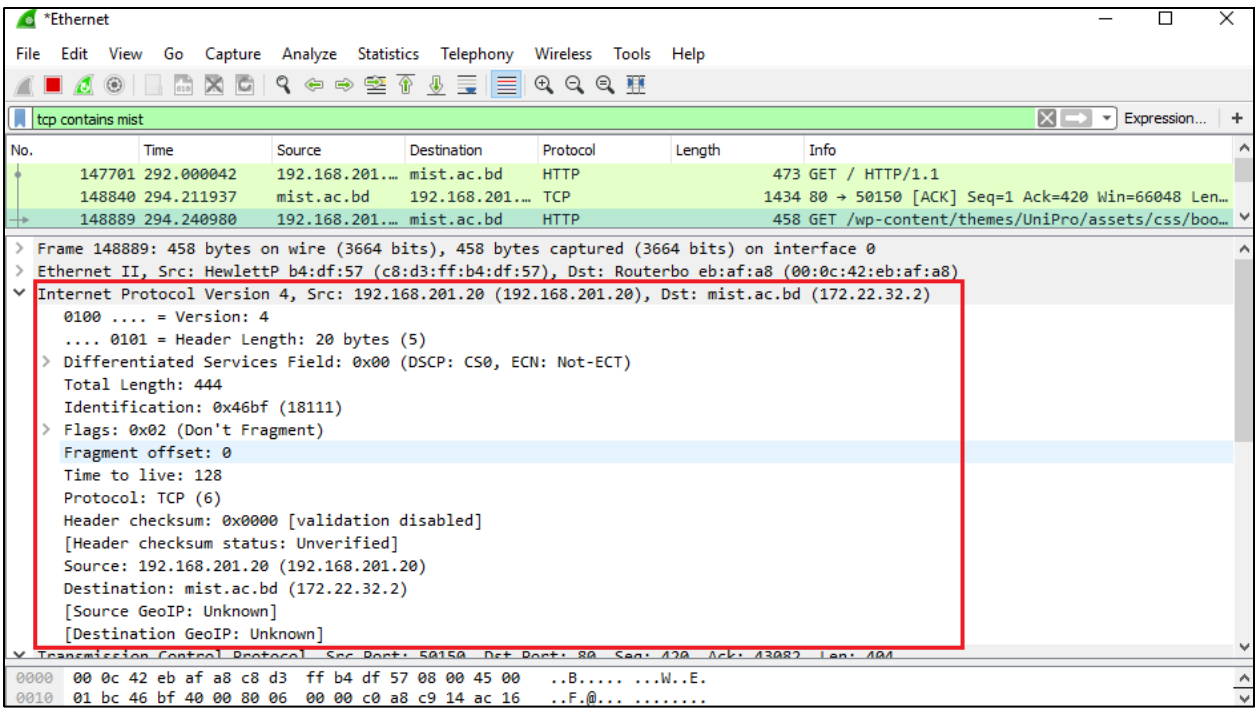
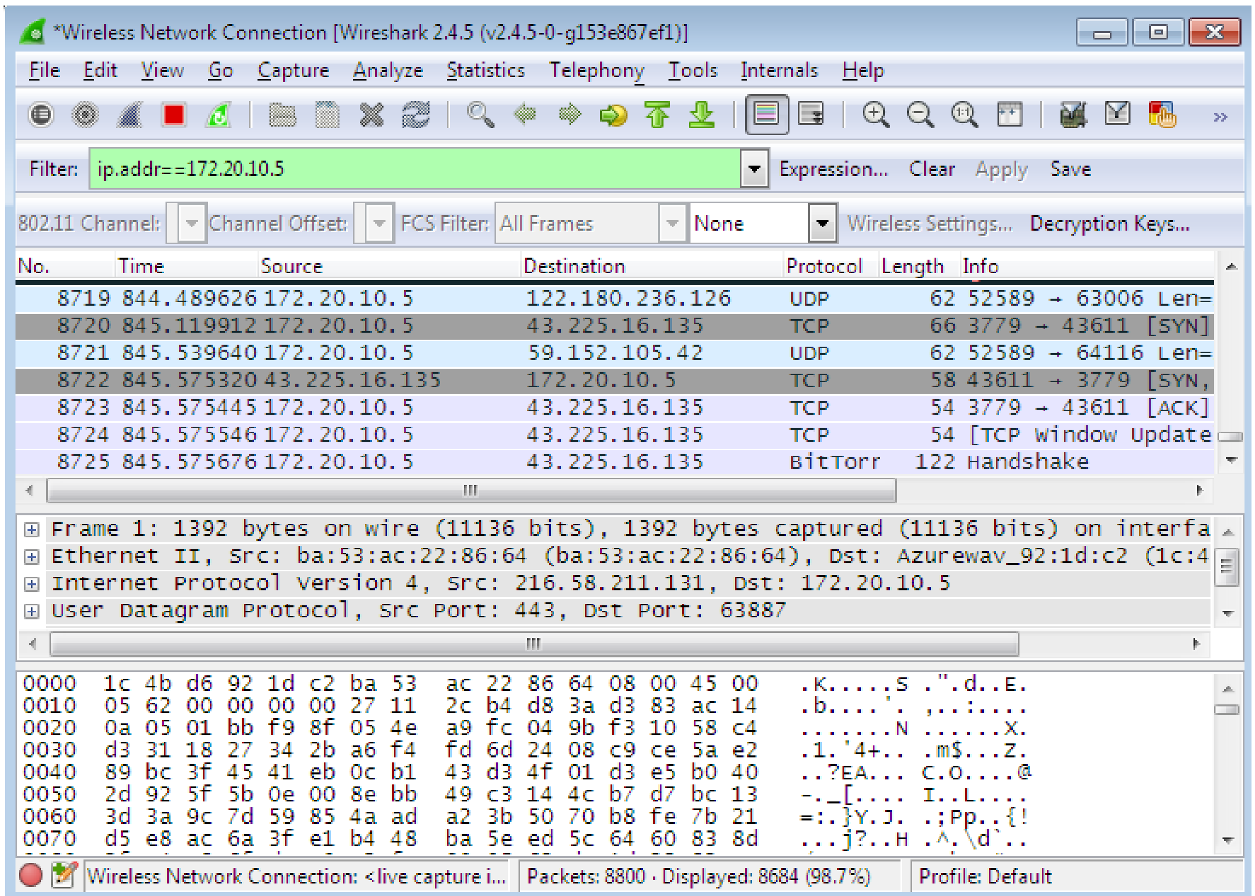


Figure-9: IPv4 Header

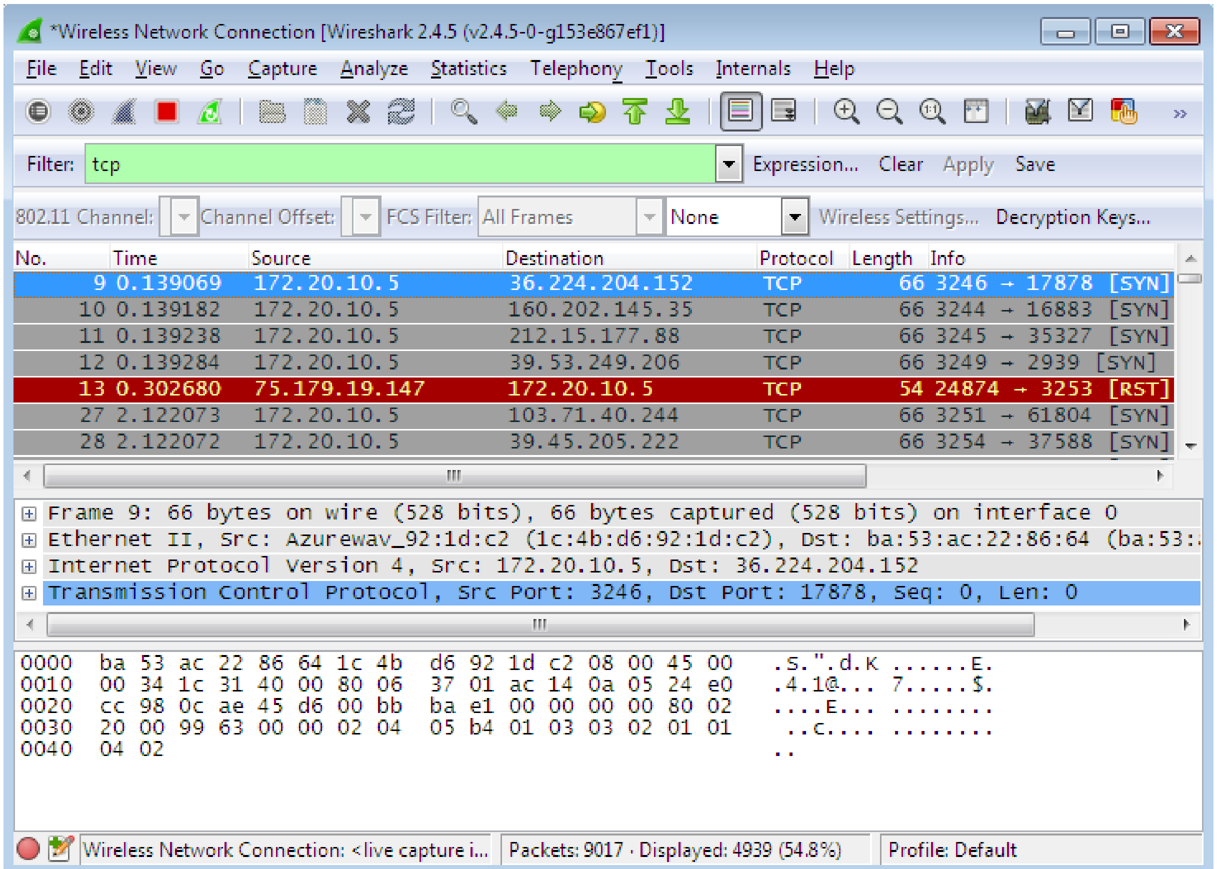
The **packet-contents window** displays the entire contents of the captured frame, in both ASCII and hexadecimal format.

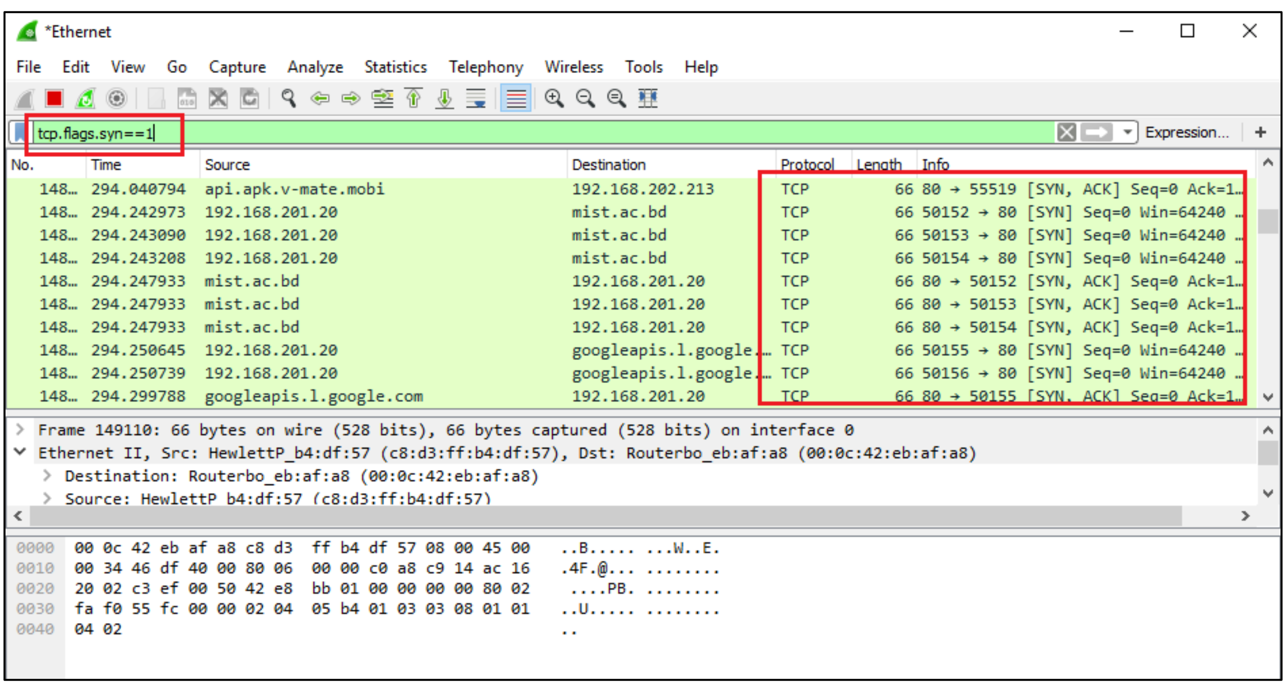
Towards the top of the Wireshark graphical user interface, is the **packet display filter field**, into which a protocol name or other information can be entered in order to filterthe information displayed in the packet-listing window (and hence the packet-headerand packet-contents windows). In the example 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:



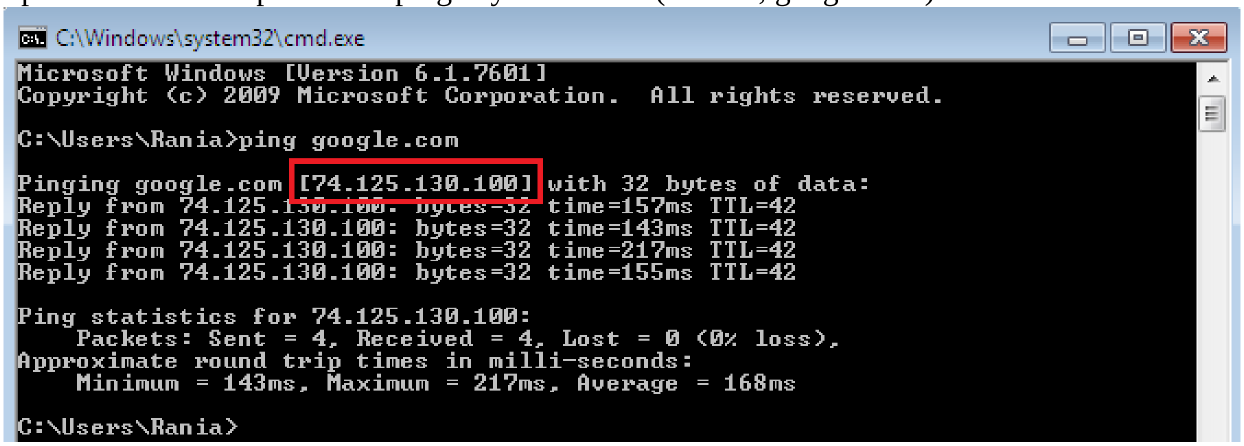
Filtering TCP Protocol



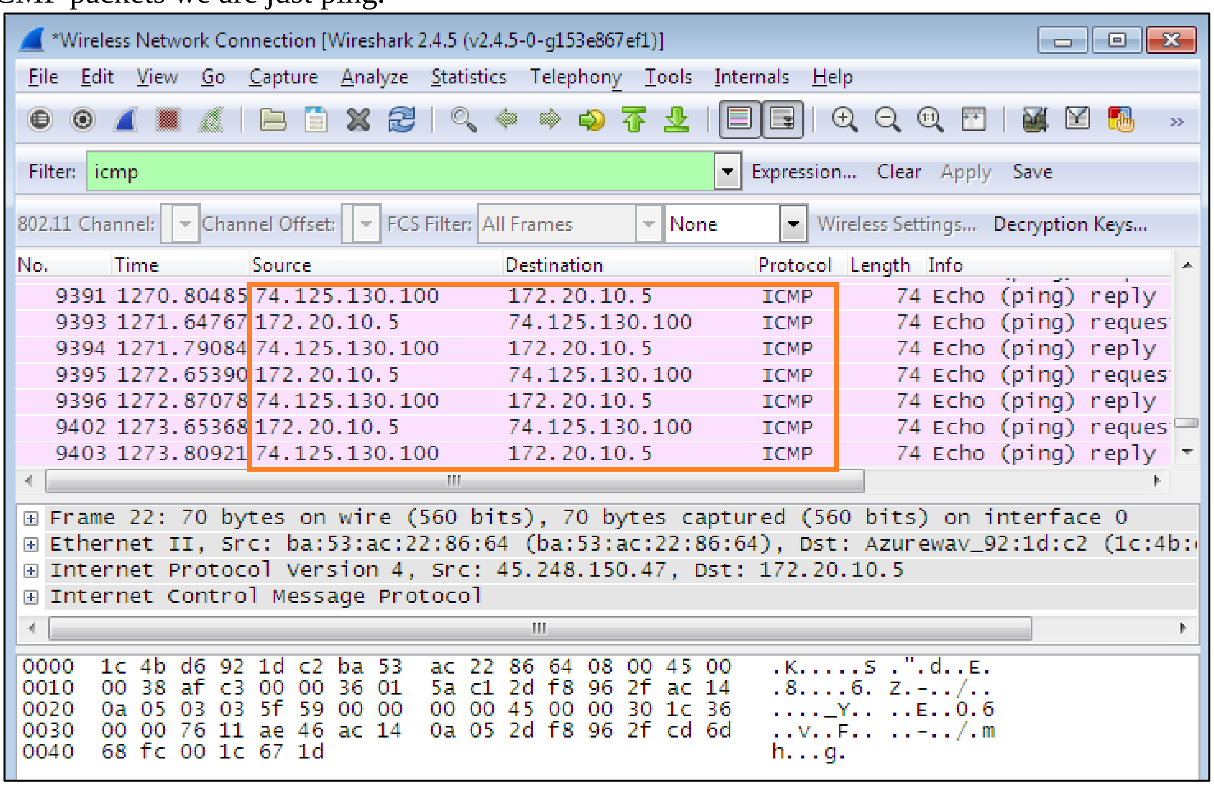


Capture ICMP Packets

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

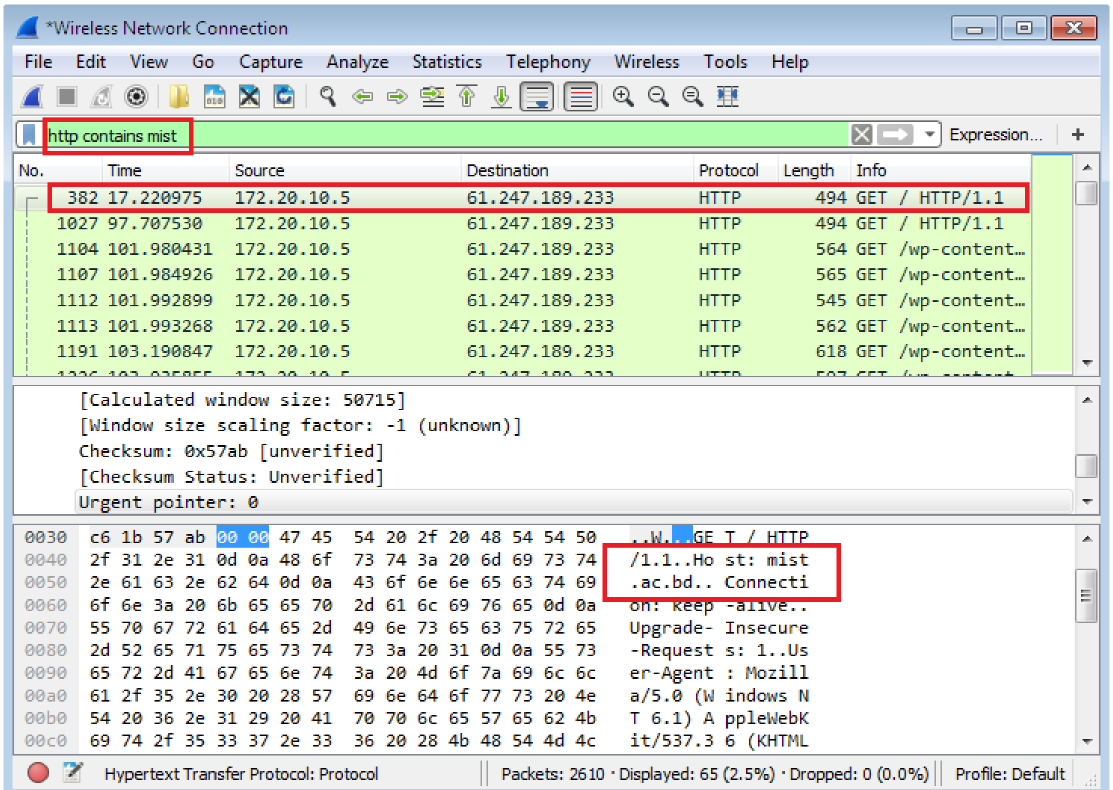


Filter wth ICMP packets and hen observe the captured packets in the wireshark. It will show the ICMP packets we are just ping.

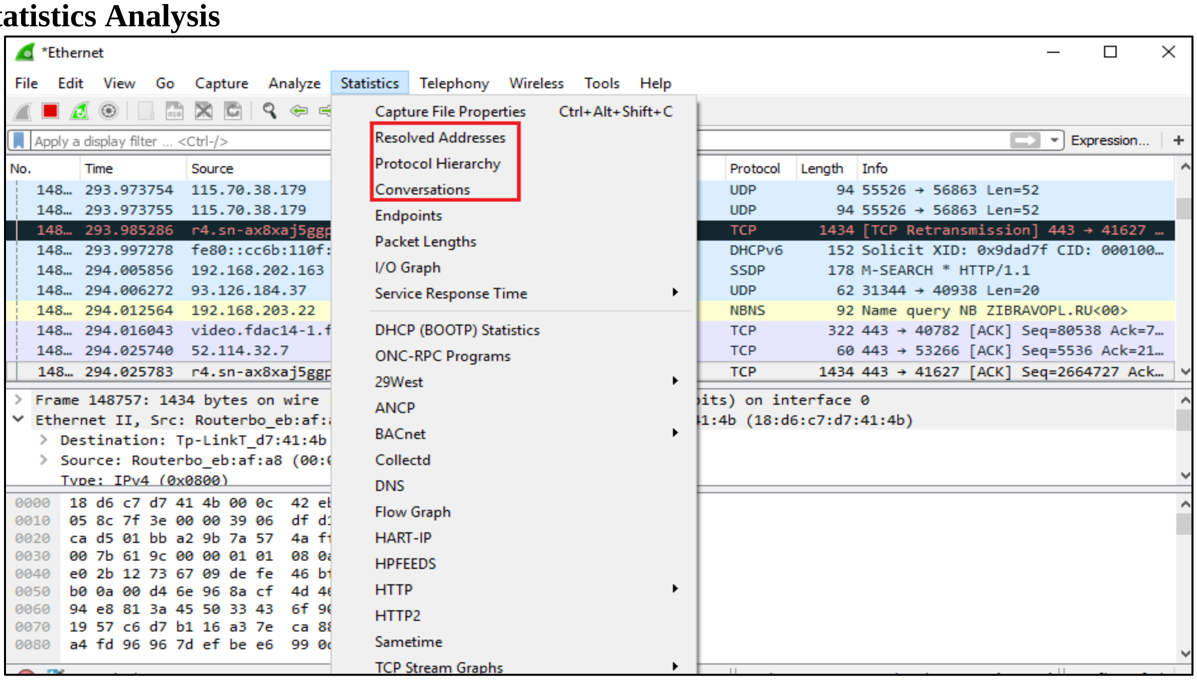


Capture HTTP packets

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

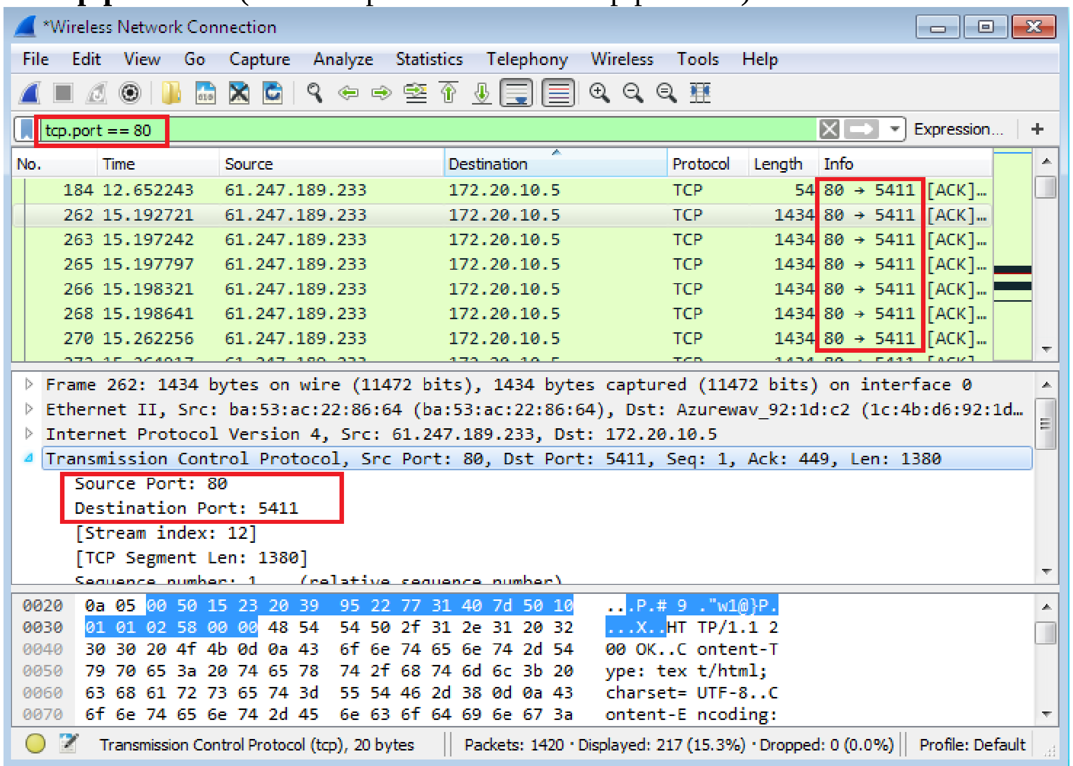


Statistics Analysis

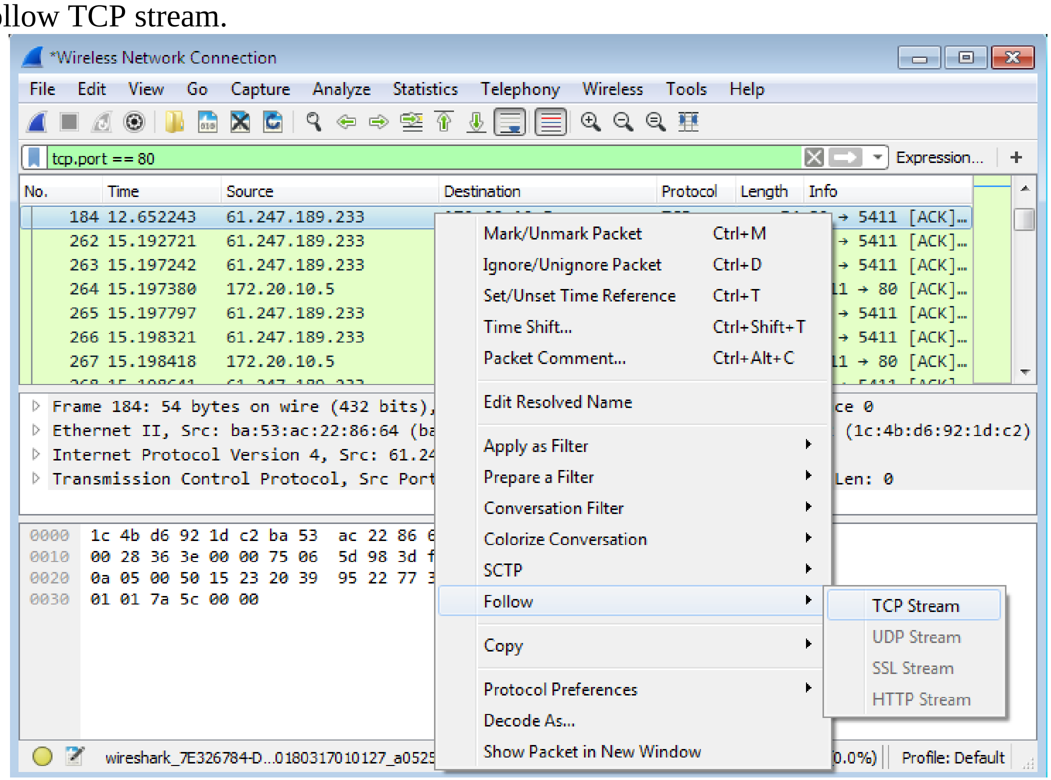


Follow TCP stream

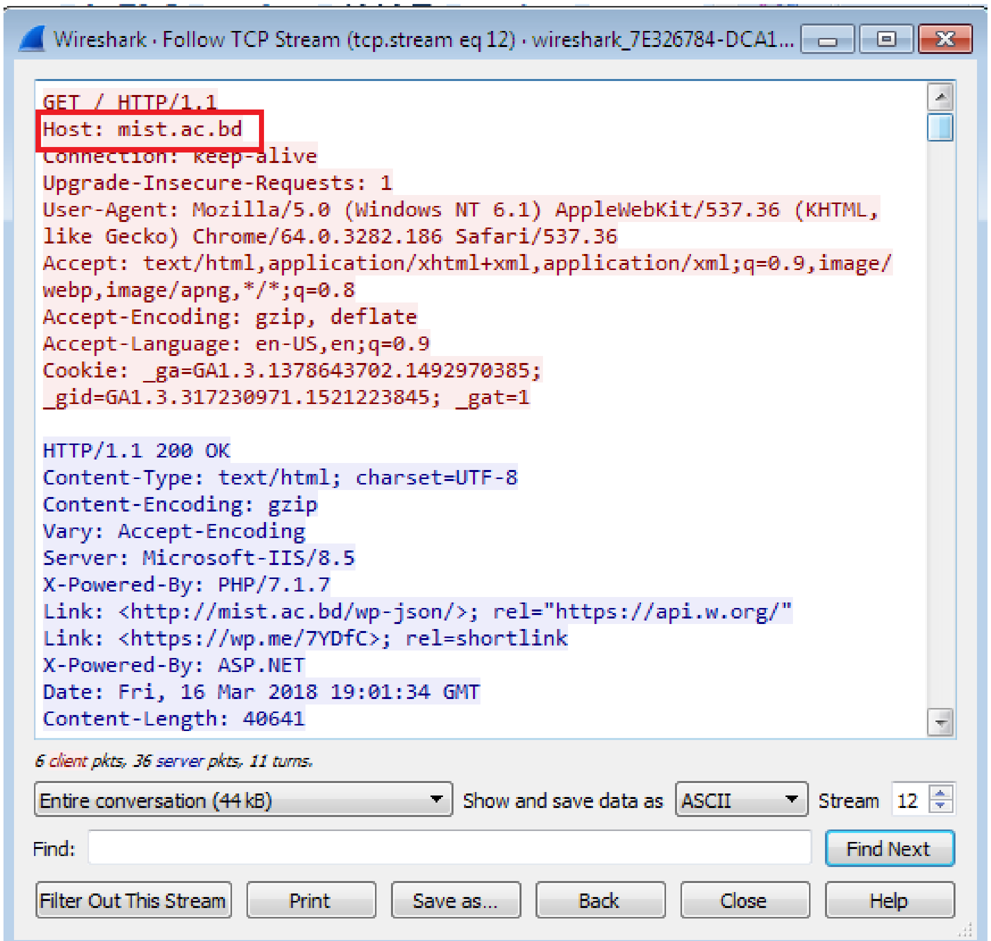
Filter with **tcp.port==80** (80 is the port number for http protocol)



Then follow TCP stream.



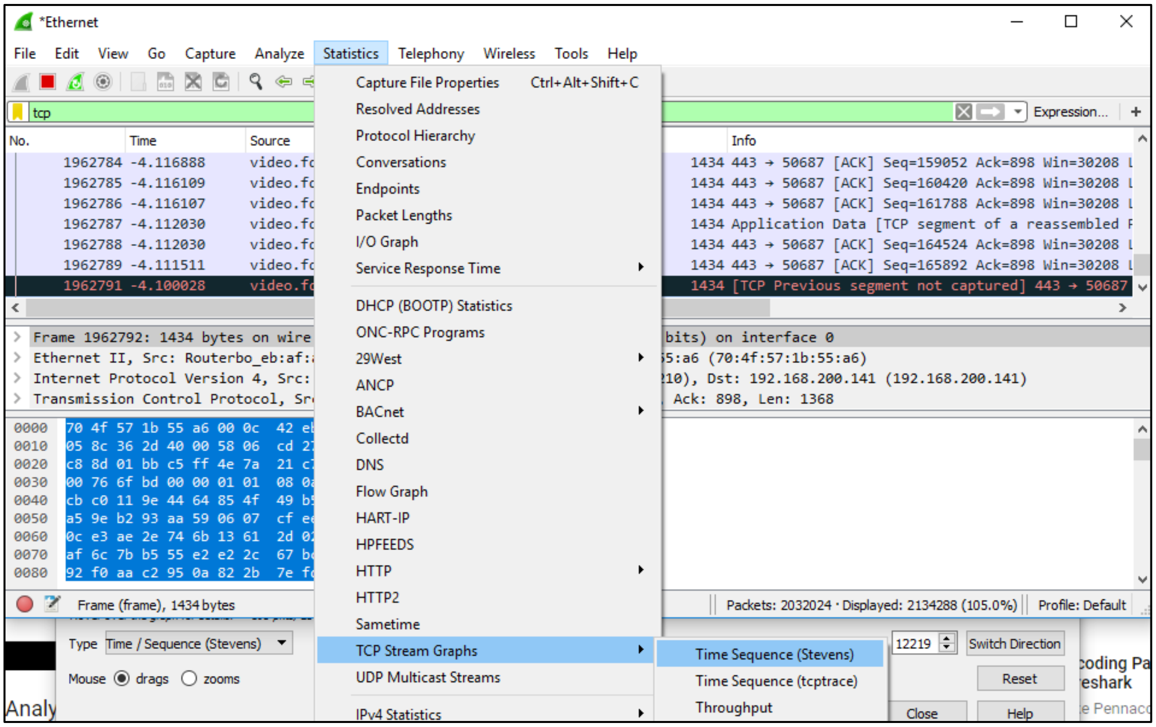
Red color is for client and Blue color ir for server side. We can change the color code.

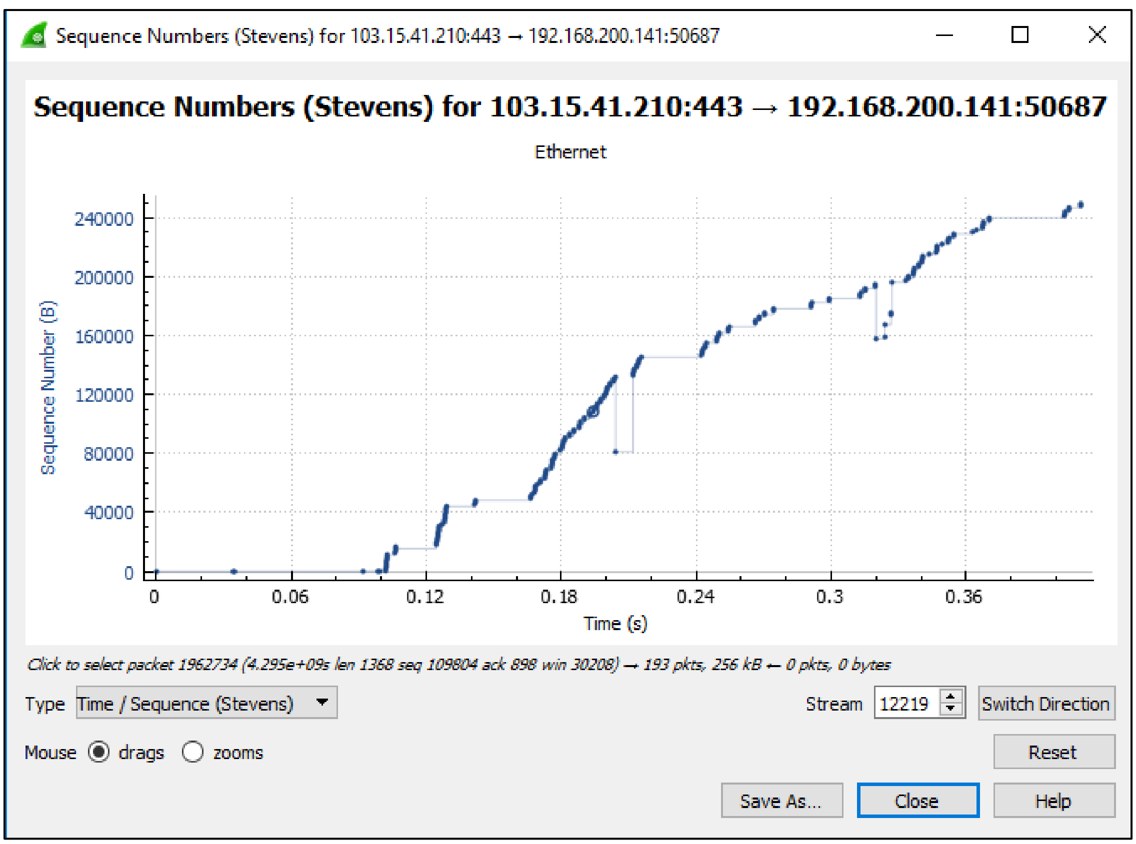


TCP Congestion Control Analysis

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).

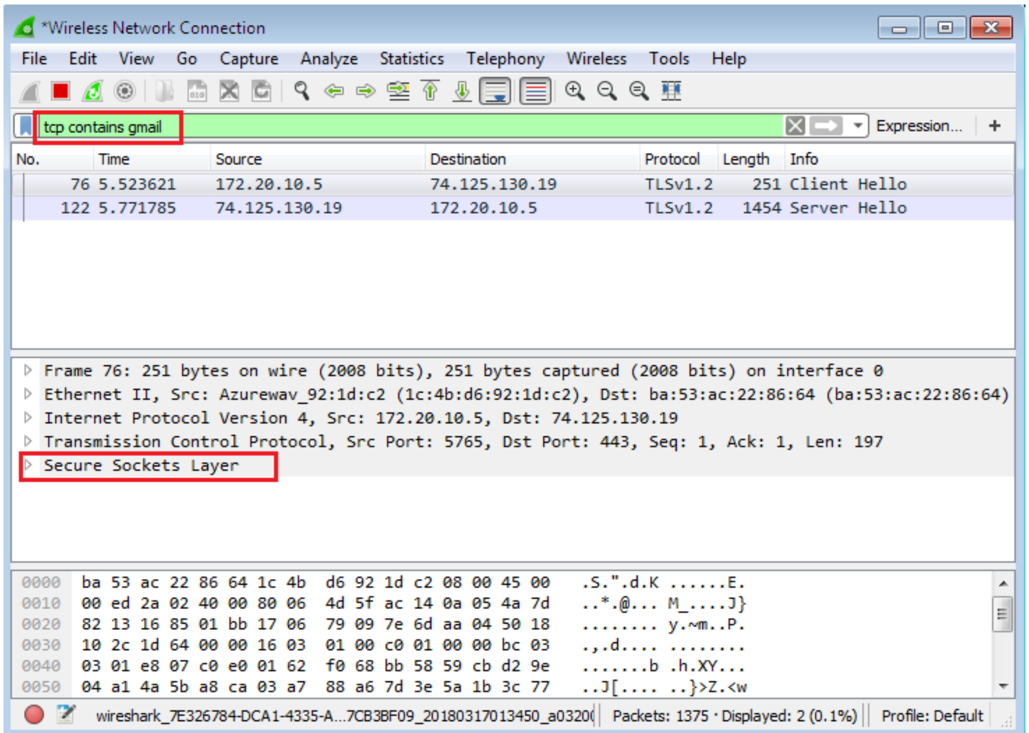
When the congestion window exceeds the slow-start threshold, ssthresh, the algorithm enters a new state, called congestion avoidance. In congestion avoidance state, as long as non-duplicate ACKs are received the congestion window is additively increased by one MSS every round-trip time.


*Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp
Expression...
Capture File Properties Ctrl+Alt+Shift+C
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
IPv4 Statistics
Time Sequence (Stevens)
Time Sequence (tcptrace)
Throughput
No. Time Source Info
1962784 -4.116888 1434 443 → 50687 [ACK] Seq=159052 Ack=898 Win=30208
1962785 -4.116109 1434 443 → 50687 [ACK] Seq=160420 Ack=898 Win=30208
1962786 -4.116107 1434 443 → 50687 [ACK] Seq=161788 Ack=898 Win=30208
1962787 -4.112030 1434 Application Data [TCP segment of a reassembled
1962788 -4.112030 1434 443 → 50687 [ACK] Seq=164524 Ack=898 Win=30208
1962789 -4.111511 1434 443 → 50687 [ACK] Seq=165892 Ack=898 Win=30208
1962791 -4.100028 1434 [TCP Previous segment not captured] 443 → 50687
Frame 1962792: 1434 bytes on wire
Ethernet II, Src: Routerbo_eb:af:
Internet Protocol Version 4, Src:
Transmission Control Protocol, Src
Frame (frame), 1434 bytes
Packets: 2032024 · Displayed: 2134288 (105.0%)
Profile: Default
Type Time / Sequence (Stevens)
Mouse drags zooms
12219 Switch Direction
Reset
Close Help

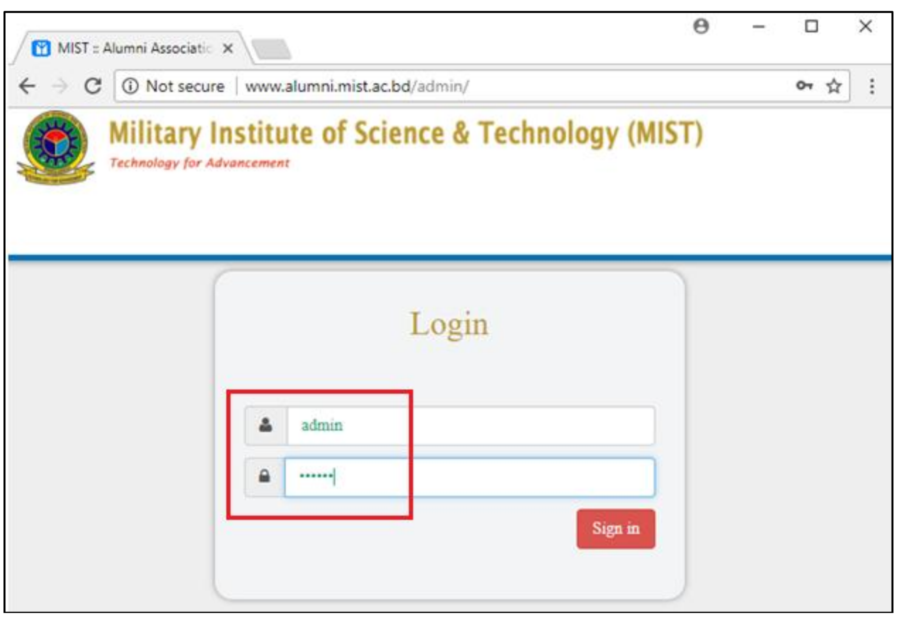

Sequence Numbers (Stevens) for 103.15.41.210:443 → 192.168.200.141:50687
Ethernet
Sequence Number (B)
240000
200000
160000
120000
80000
40000
0
0 0.06 0.12 0.18 0.24 0.3 0.36
Time (s)
Click to select packet 1962734 (4.295e+09s len 1368 seq 109804 ack 898 win 30208) → 193 pkts, 256 kB ← 0 pkts, 0 bytes
Type Time / Sequence (Stevens)
Stream 12219 Switch Direction
Mouse drags zooms
Reset
Save As... Close Help

Sniffing Password

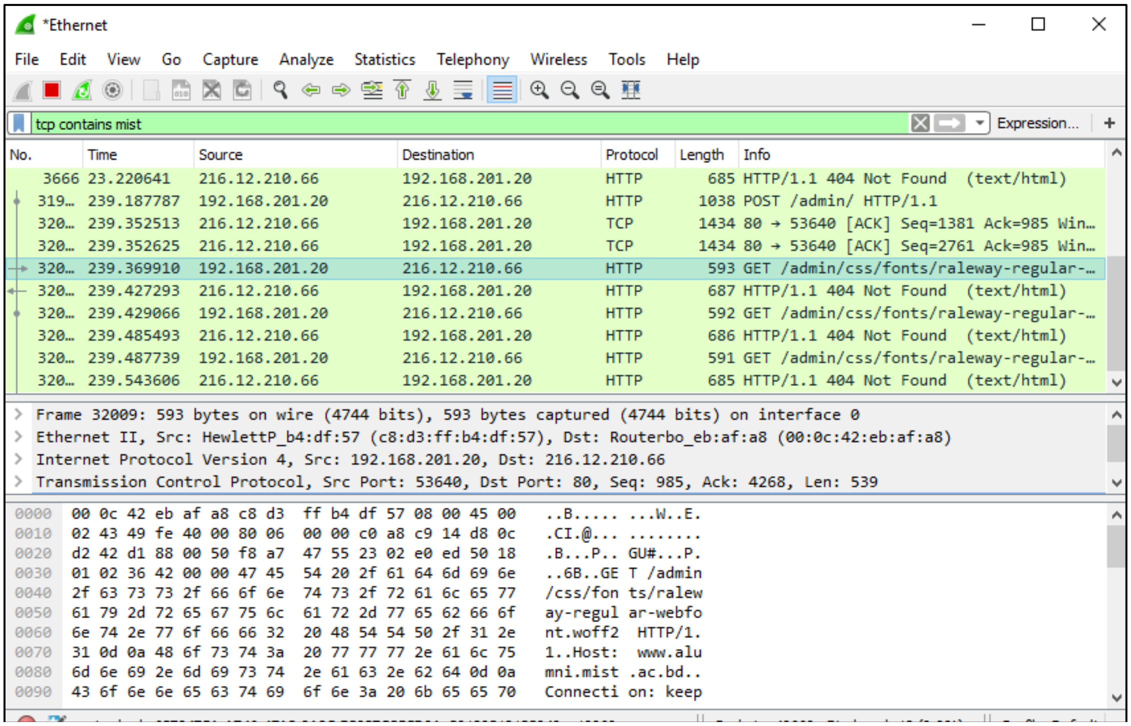
If we open **gmail.com** in a browser. Login and try to follow the TCP stream for sniffing username and password, we cannot. Because Gmail use SSL (Secured Socket Layer).

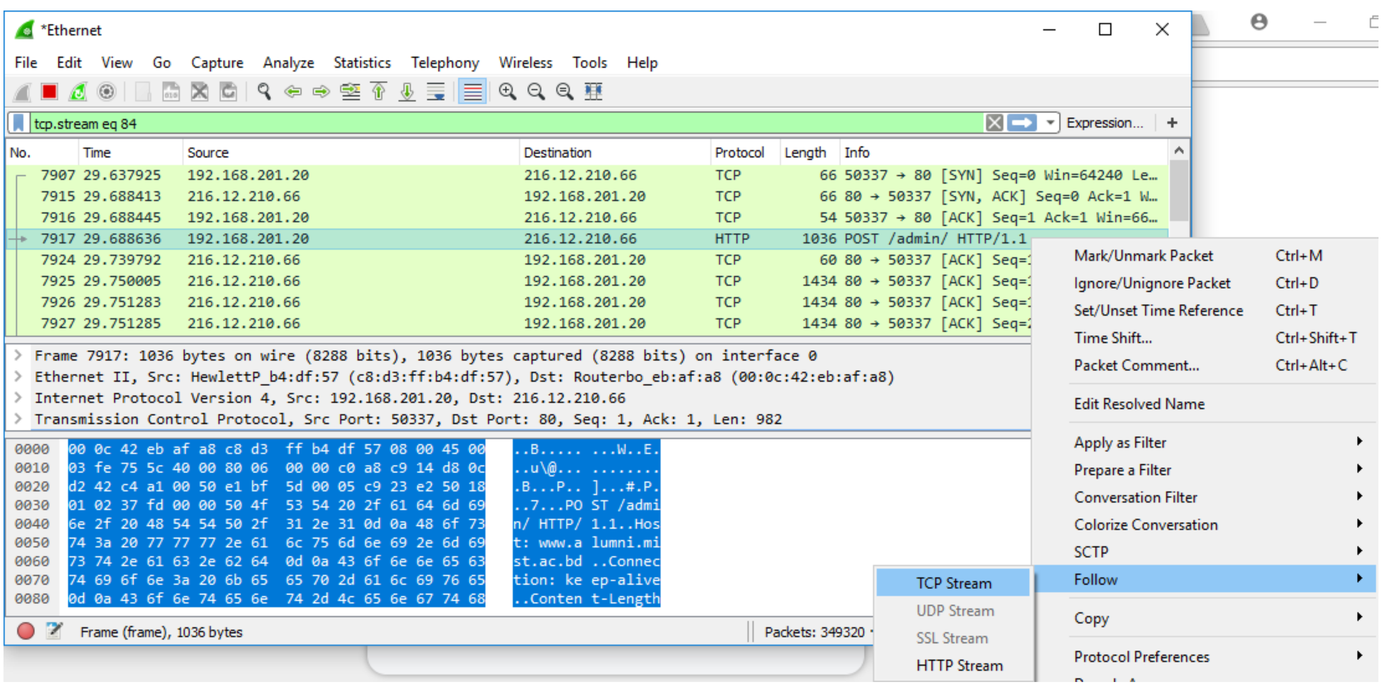


Now, open **www.alumni.mist.ac.bd/admin/** in a browser. Login with **username: admin** and **password: 123456**. There should be a message “Username and Password combination error!” because the password in incorrect.



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





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