



WIRESHARK

WIRESHARK

BASICS OF WIRESHARK

ABHISHEK GUATAM
LUCIDEUS

INTRODUCTION

Wireshark is a **network packet analyzer**. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible.

Wireshark is perhaps one of the best open source packet analyzers available today.

Features of WIRESHARK

Available for UNIX and Windows.

Capture live packet data from a network interface.

Open files containing packet data captured with tcpdump/WinDump, Wireshark, and a number of other packet capture programs.

Import packets from text files containing hex dumps of packet data.

Display packets with very detailed protocol information.

Save packet data captured.

Export some or all packets in a number of capture file formats.

Filter packets on many criteria.

Search for packets on many criteria.

Colorize packet display based on filters.

Create various statistics.

Wireshark captures packets and lets you examine their contents.

Live capture from many different network media

Import files from many other capture programs

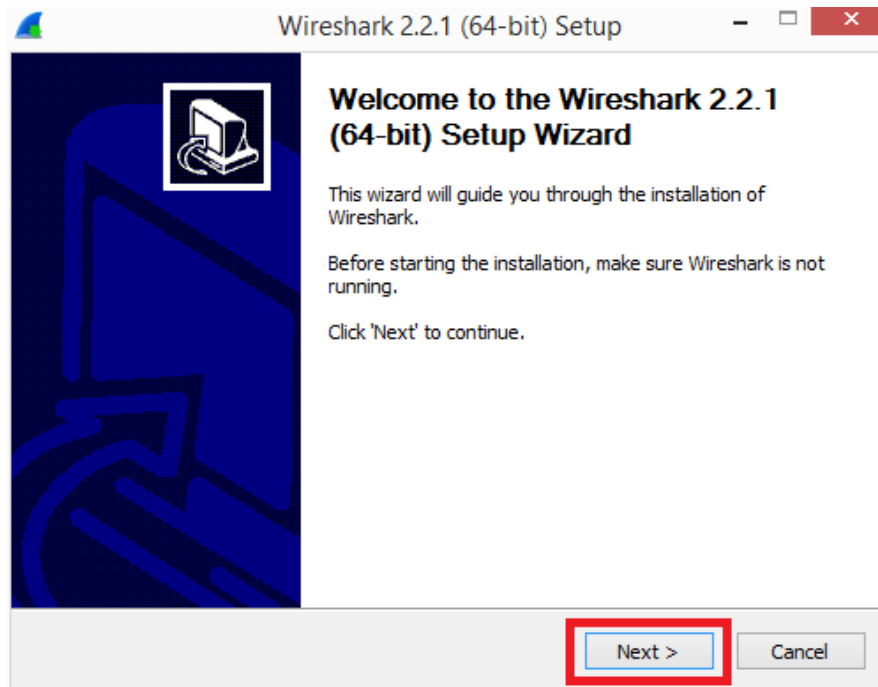
Export files for many other capture programs

Many protocol dissector

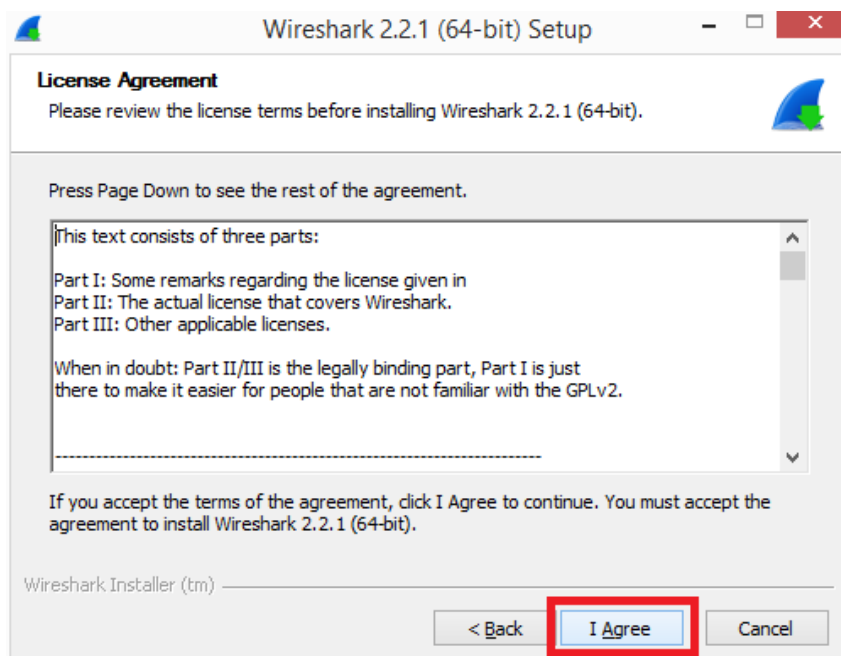
Open Source Software

INSTALLING WIRESHARK

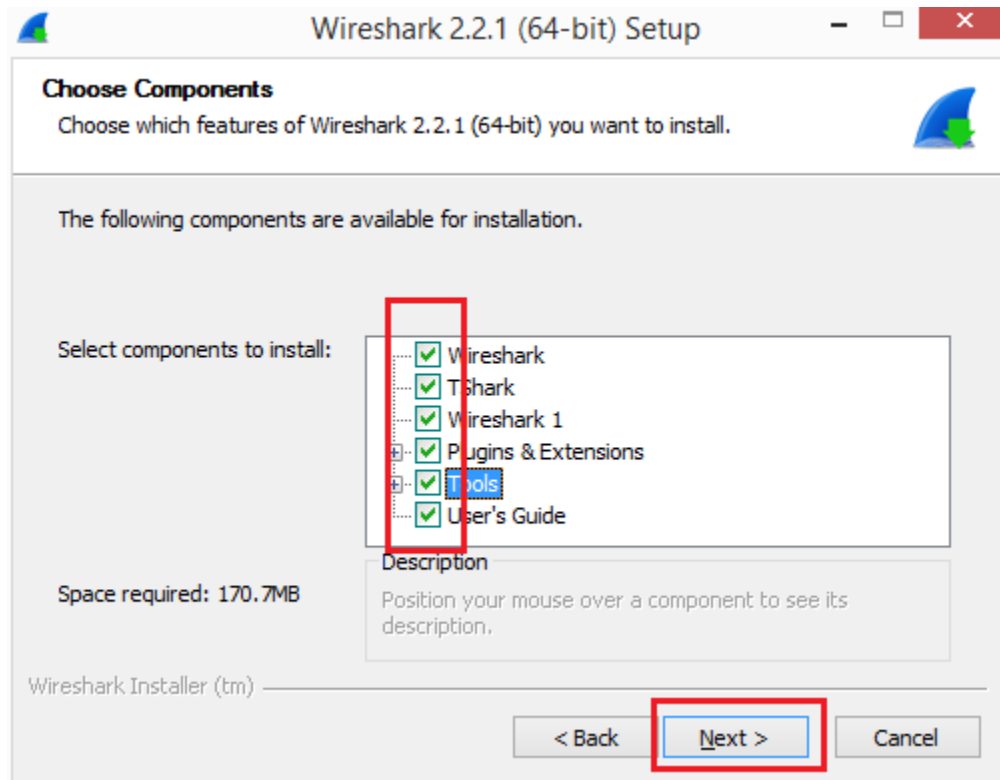
STEP 1: Double click on the downloaded .exe file of Wireshark. Click yes if UAC is prompted and then click “NEXT”.



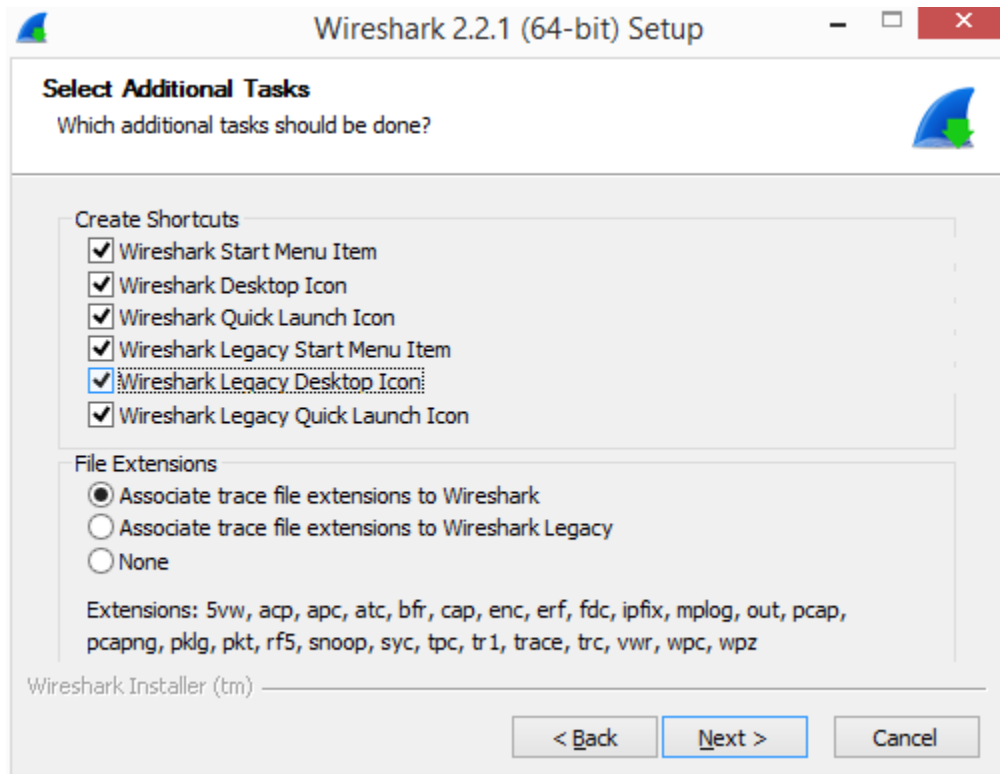
STEP 2: Accept the License Agreement by clicking “I Agree”



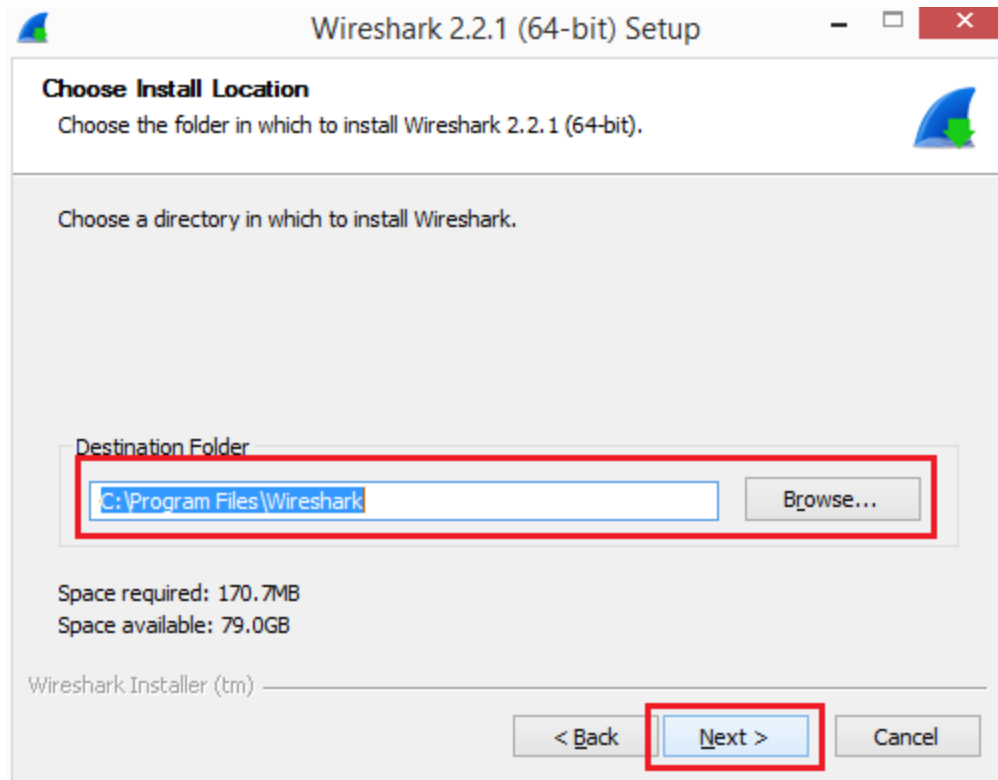
STEP 3: Tick all the check box and then click "NEXT".



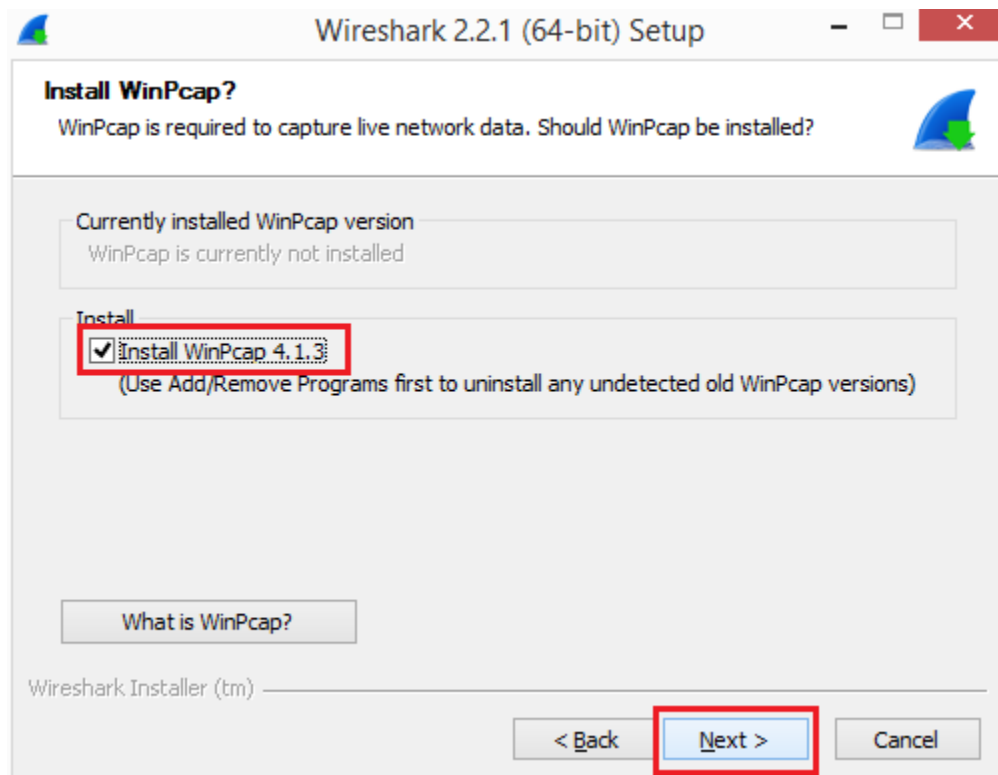
STEP 4: In the Following Screen select the options according to your own choice.



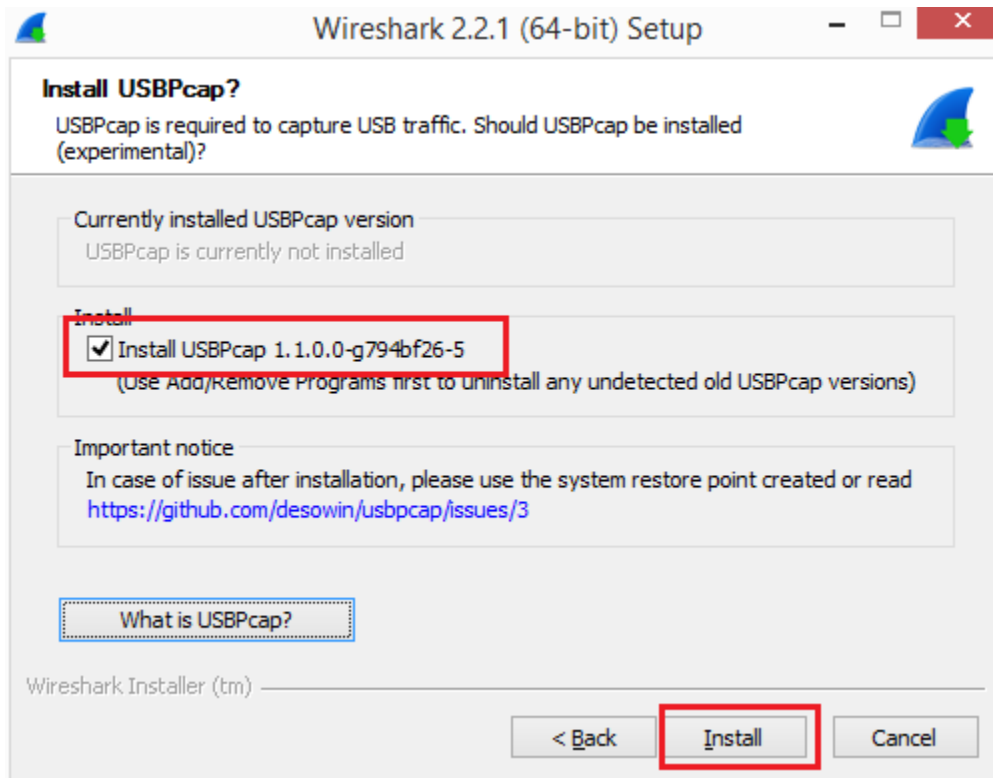
STEP 5: Select the directory in which you want to install but it is recommended to install the program in the default directory.



STEP 6: CLICK "Install WinPcap". winPcap is the application needed for capturing purpose. (For windows users only)



STEP 7: CHECK install “Install USBPcap”. USBPcap is an open-source USB sniffer for Windows.



OVERVIEW OF INTERFACE

The Wireshark Network Analyzer [Wireshark 2.2.1 (v2.2.1-0-ga6fbd27 from master-2.2)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help Main Menu Bar Appear

commonly used tools

Filter: Expression... Clear Apply Save Filtering Specific Packet

WIRESHARK The World's Most Popular Network Protocol Analyzer
Version 2.2.1 (v2.2.1-0-ga6fbd27 from master-2.2)

Capture

Interface List
Live list of the capture interfaces (counts incoming packets)

Start
Choose one or more interfaces to capture from, then **Start**

Ethernet
Wi-Fi 2
Wi-Fi
Local Area Connection* 3
Bluetooth Network Connection
USBPCap1
USBPCap2
Cisco remote capture
Random packet generator
SSH remote capture

Capture Options
Start a capture with detailed options

Capture Help

Files

Open
Open a previously captured file

Open Recent:

Sample Captures
A rich assortment of example capture files on the wild

help you open your previously saved cap files

takes you to websites having sample captures

List Of available interfaces on which we can capture. This section may be different for everybody

Help Section

Online

Website
Visit the project's website

User's Guide
The User's Guide (online version)

Security
Work with Wireshark as securely as possible

official website link

User Guide (Online version)

Ready to load or capture No Packets Profile: Default

Capturing from Wi-Fi 2 [Wireshark 2.2.1 (v2.2.1-0-ga6fbd27 from master-2.2)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
274	15.695595	89.146.224.58	192.168.1.15	TCP	54	80-56999 [ACK] Seq=1 Ack=369 win=30336 Len=0
275	15.695596	89.146.224.58	192.168.1.15	HTTP	856	HTTP/1.1 503 Service Temporarily Unavailable (text/html)
276	15.695596	89.146.224.58	192.168.1.15	TCP	1514	[TCP Previous segment not captured] 80-56860 [ACK] Seq=204401 Ack=1 win=237 Len=1460
277	15.695596	89.146.224.58	192.168.1.15	TCP	1514	80-56860 [ACK] Seq=205861 Ack=1 win=237 Len=1460
278	15.696840	Netgear_80:03:00	Tp-LinkT_0b:65:1c	ARP	42	192.168.1.1 is at c4:04:15:80:03:00
279	15.696862	192.168.1.15	89.146.224.58	TCP	66	[TCP Dup ACK 272#1] 56860-80 [ACK] Seq=1 Ack=201481 win=256 Len=0 SLE=204401 SRE=207321
280	15.697311	192.168.1.15	89.146.224.58	TCP	54	56999-80 [RST, ACK] Seq=369 Ack=803 win=0 Len=0
281	16.133599	89.146.224.58	192.168.1.15	TCP	1514	[TCP Retransmission] 80-56860 [ACK] Seq=201481 Ack=1 win=237 Len=1460
282	16.181968	192.168.1.15	89.146.224.58	TCP	66	56860-80 [ACK] Seq=1 Ack=202941 win=256 Len=0 SLE=204401 SRE=207321
283	17.021138	89.146.224.58	192.168.1.15	TCP	1514	[TCP Retransmission] 80-56860 [PSH, ACK] Seq=202941 Ack=1 win=237 Len=1460
284	17.072172	192.168.1.15	89.146.224.58	TCP	54	56860-80 [ACK] Seq=1 Ack=207321 win=256 Len=0
285	17.524352	89.146.224.58	192.168.1.15	TCP	1514	80-56860 [ACK] Seq=207321 Ack=1 win=237 Len=1460

Frame 1: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0

Ethernet II, Src: Netgear_80:03:00 (c4:04:15:80:03:00), Dst: Tp-LinkT_0b:65:1c (18:a6:f7:0b:65:1c)

Internet Protocol Version 4, Src: 89.146.224.58, Dst: 192.168.1.15

Transmission Control Protocol, Src Port: 80, Dst Port: 56860, Seq: 1, Len: 1460

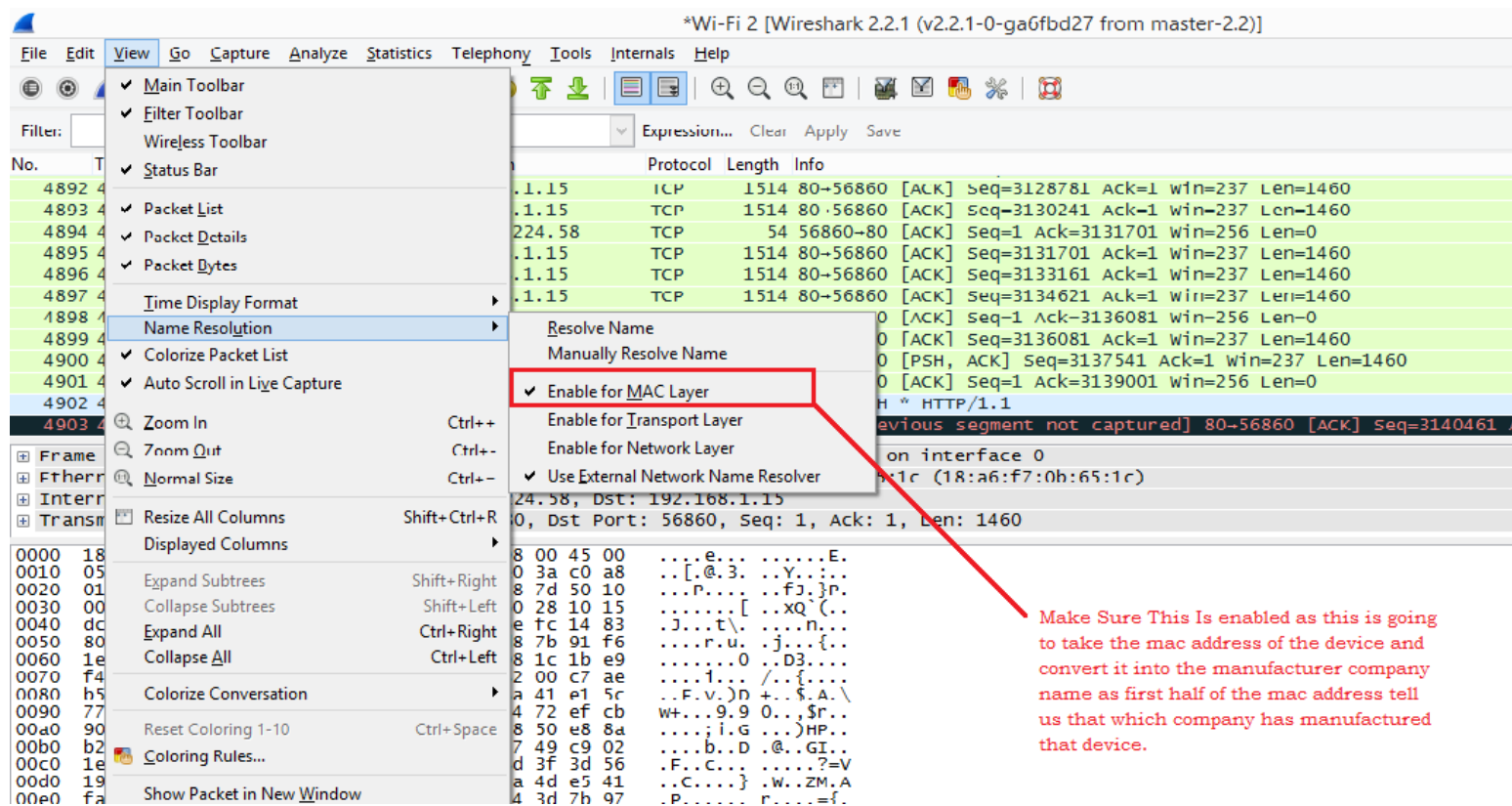
0000 18 a6 f7 0b 65 1c c4 04 15 80 03 00 08 00 45 00e.....E.
0010 05 dc 5b d4 40 00 33 06 ea c3 59 92 e0 3a c0 a8 ...[.@.3..Y.....
0020 01 0f 00 50 de 1c 1f 1e b1 8a 66 4a 18 7d 50 10 ...P.....fj..P.
0030 00 ed 12 fe 00 00 14 5b cd b0 78 51 60 28 10 13[.xq(.
0040 dc 4a 8b e3 15 74 5c 8d be fa 98 84 6e fc 14 83 ...t..v.....
0050 80 88 15 bb 72 f5 75 90 f3 6a c2 cf d8 7b 91 f6r.u..j.....
0060 1e 9d 80 af e9 b1 be 30 e4 c0 44 33 98 1c 1b e90..D3....
0070 f4 a1 b7 ef 69 e8 2e 81 2f f9 f3 7b 82 00 c7 aei..../{....
0080 b5 a9 45 d8 76 ab 29 44 2b b2 ad 24 8a 41 e1 5c ...E.v.D+...\$.A..
0090 77 2b 99 d9 c8 39 10 39 30 d8 c3 2c 24 72 ef cb wh...9.0...\$.r..
00a0 90 8f fa b2 3b 69 9c 47 93 8e 95 29 48 50 e8 8ai.G...HP...
00b0 02 b4 00 86 02 b6 09 44 c4 07 1b 47 49 c9 02b..b...e...
00c0 1e 46 c4 b4 63 d7 84 87 0e 8d e8 d8 8d 3f 3d 56 ...F.C.....?..V
00d0 19 ea 43 ea eb 2e bf 7d b6 57 fd 91 5a 4d e5 41 ...C.....}..W..ZM.A

Shows Information About our packets in human readable form. we can hide this by going to view and untick packet details

This is the BITS VIEW. This is actually how our routers and netcards are going to see the packets. it is not in human readable form. We can hide this by going to View and untick Packet byte

All Packets Pop Up Here We can hide this by going to view and untick packet list

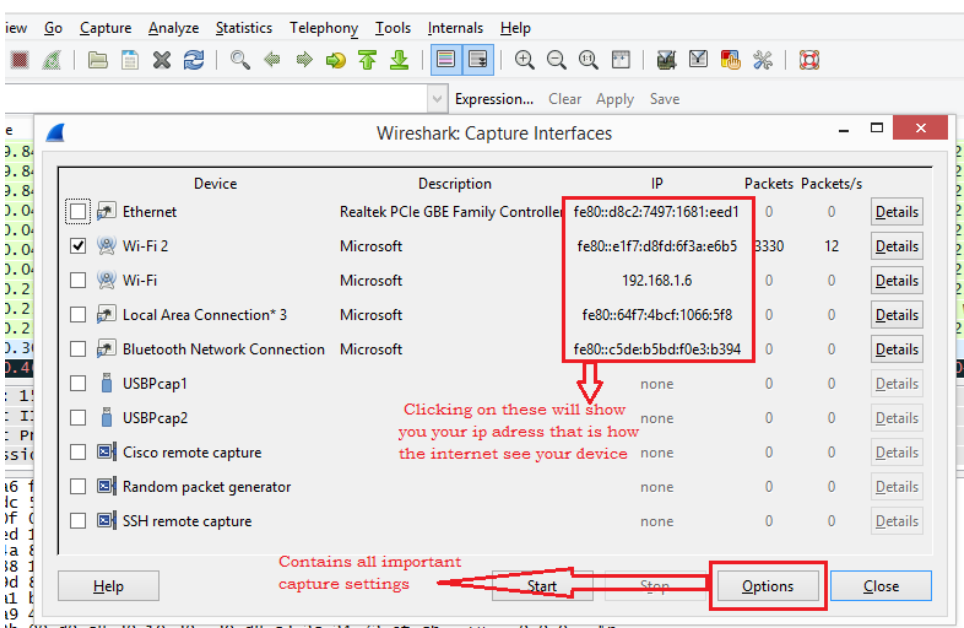
NOTE WE CAN CHANGE THE LAYOUT OF WIRESHARK BY GOING TO EDIT AND THEN TO PREFERENCES AND SET IT ACCORDING TO OUR OWN WISH.



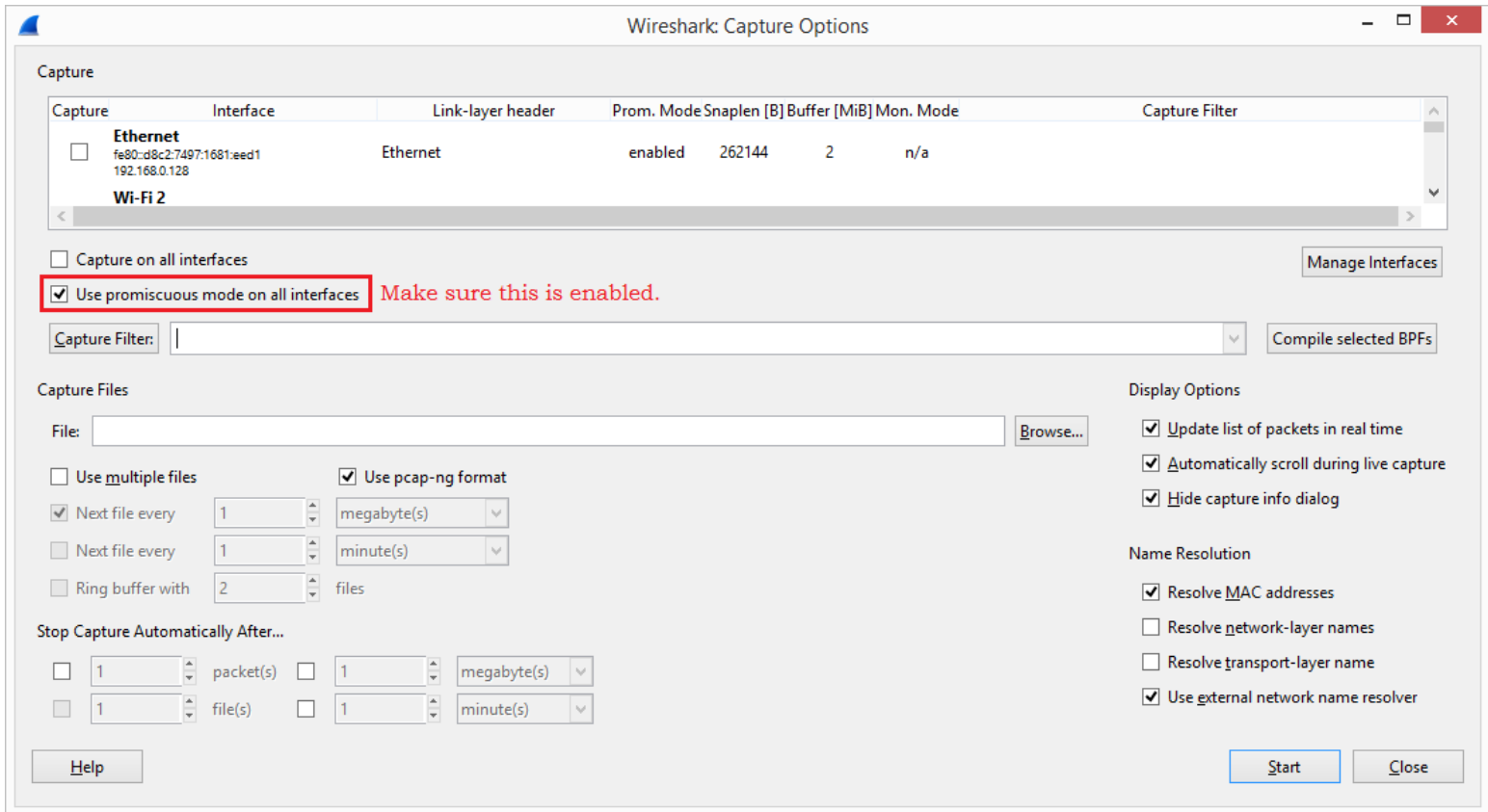
Capture Options

These settings are very important as it tells the wireshark what we want to listen on the network.

Step 1: Goto to capture options and click on interfaces the keyboard shortcut for this is ctrl+I. A screen similar to following screen will popup.




Step 2: now click on options. It will give more capture options and will give us more control.




Make sure listening is enabled in promiscuous mode. Now let's understand what promiscuous mode is. Let's suppose you went to a coffee shop and you wanted to sniff everyone's traffic although that is illegal and you should never do it, but you start capturing and something weird started you noticed that you were only able to view your own traffic. This happened because all the packets which came to our machine which was not meant for us were ignored by our machine and this is how it builds. So what promiscuous mode does is that it enables the device to listen to everyone else's traffic but sometimes the promiscuous mode is default blocked in our interface device so for this we need to buy a device which supports promiscuous mode.

TOOLBAR ICONS



 : This shows the interface list

 : Stop Capture

 : This shows capture options.

 : Refresh or Restart

 : Start Capture



: Open a capture File



: Save the capture File



: Close the capture file



: Reload the current capture file



: Find a packet



: Go back in packet History



: Go forward in packet History



: Goto packet with number(Packet Id)



: Goto first packet



: Goto Last packet



:Edit preference



: Show some help



: Colorize packet list



: Auto scroll packet list in live capture



: Zoom in



: Zoom out



: Zoom 100%



: Resize all columns



: Edit capture file



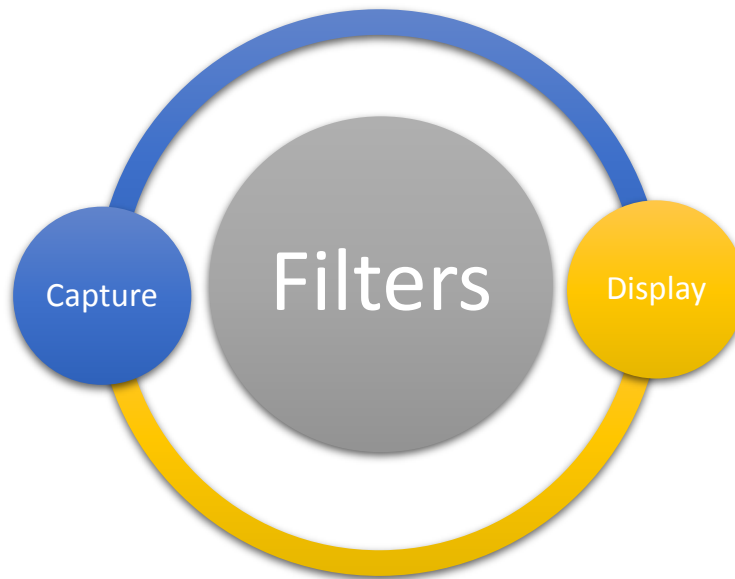
: Edit display filter



: Edit packet coloring rule

FILTERS

it is a way to filter out our packets. When we do capturing the packets we get lots of packets out of which we might not worry about a lot of packets like all UDP Packets and Some TCP Packets so here filter proves to be a great feature for us . There are two different types of filters:



- 1) Display Filters: This is a display filter.
Example we want to view only the http
Packets so we can type http and press enter and then it will only display the http packets.

Filter:

Filter: **http** type the filter you want to apply. Expression... Clear Apply Save

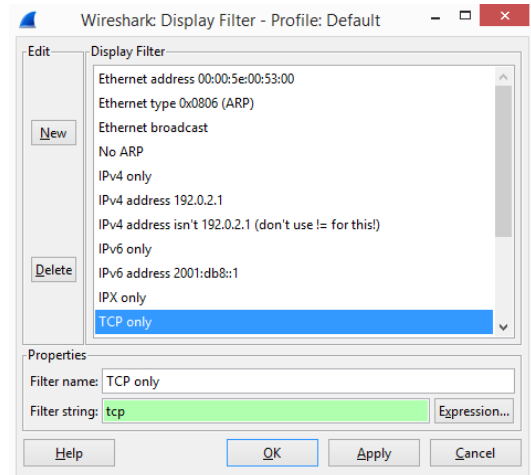
No.	Time	Source	Destination	Protocol	Length	Info
14	0.601624	89.146.224.58	192.168.1.15	TCP	1514	80->56860 [ACK] Seq=8761 Ack=1 win=237 Len=1460
109	6.981822	192.168.1.15	89.146.224.58	HTTP	422	GET /download/vm/OpenVAS/OpenVAS-8-DEMO-1.0.ova HTTP/1.1
113	7.154316	89.146.224.58	192.168.1.15	HTTP	856	HTTP/1.1 503 Service Temporarily Unavailable (text/html)
269	15.244591	192.168.1.15	89.146.224.58	HTTP	422	GET /download/vm/OpenVAS/OpenVAS-8-DEMO-1.0.ova HTTP/1.1
275	15.695596	89.146.224.58	192.168.1.15	HTTP	856	HTTP/1.1 503 Service Temporarily Unavailable (text/html)
312	19.748487	89.146.224.58	192.168.1.15	TCP	1514	80->56860 [ACK] Seq=230681 Ack=1 win=237 Len=1460
398	37.144960	192.168.1.15	89.146.224.58	HTTP	422	GET /download/vm/OpenVAS/OpenVAS-8-DEMO-1.0.ova HTTP/1.1
401	37.419702	89.146.224.58	192.168.1.15	HTTP	856	HTTP/1.1 503 Service Temporarily Unavailable (text/html)
591	45.956791	192.168.1.15	89.146.224.58	HTTP	422	GET /download/vm/OpenVAS/OpenVAS-8-DEMO-1.0.ova HTTP/1.1
650	49.164563	89.146.224.58	192.168.1.15	HTTP	856	HTTP/1.1 503 Service Temporarily Unavailable (text/html)
656	49.781323	77.234.43.34	192.168.1.15	HTTP	234	HTTP/1.1 200 OK (application/octet-stream)
657	49.783136	192.168.1.15	77.234.43.34	HTTP	344	GET /R/A28KIGRkMDY5OWRiYzJlZTQxZTY4MGZhNmNlNTdmODFmNGQzEgQ

- 2) Capture Filters: It is in the capture settings. If we put http in the capture filter it would not even log any other packet except http during the capturing process.

Capture Filter: Compile selected BPFs

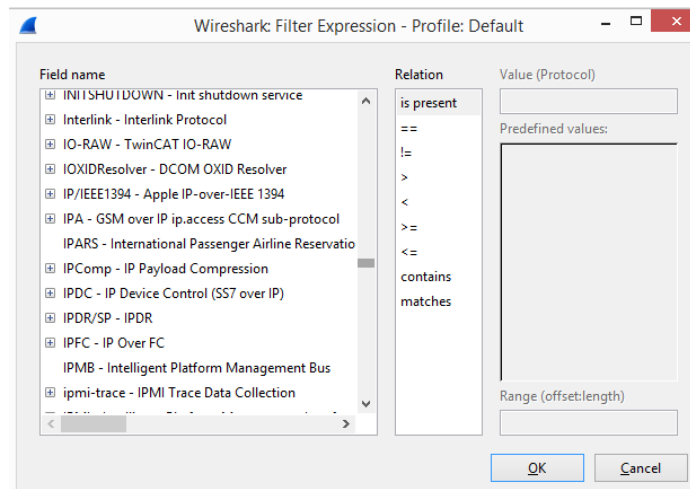
NOTE: if we click on the filter button in the left most corner we get a list of most common filters and we can apply any of it or add our own filter in that list. And one more point to note is that whenever we are typing any display filter that is not a valid filter it will show in red color.

Now suppose we want to filter out only GET packets then we can proceed as follow by typing `http.request.method == "GET"`



Now lets understand it we are saying we want to filter http packet which are having a request and we want to filter it by method and that method should be GET.

IF we click on expression button it will show us different types of filter we can apply by the use of expressions.



We can also combine two or more filter like we want to view the http GET AND POST method packets we can type

`(http.request.method == "GET") || (http.request.method == "POST")` those familiar to programming will understand it better. This is a way of saying show me all packets which are using GET or POST. We can also use `&&` which will do a logical and between the two filters that is it will display only those packets which satisfy both the conditions, although if we use it here in this example it won't make any sense. We can also save our filters by clicking the save for future use.

No.	Time	Source	Destination	Protocol	Length	Info
-----	------	--------	-------------	----------	--------	------

INTERFACE CONTROLS

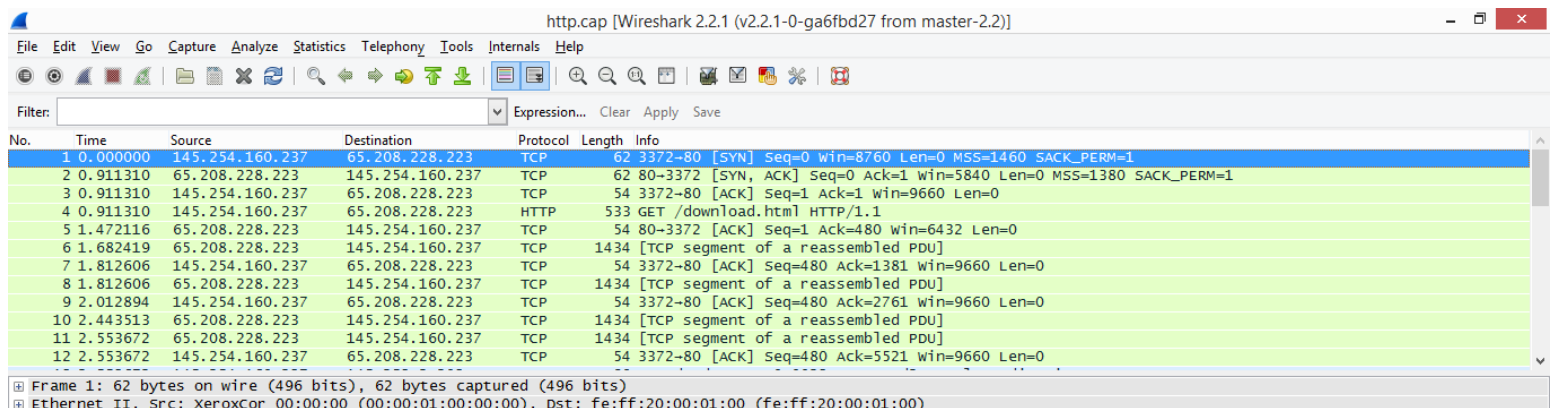
- 1) The No. tells us the packet number it acts like a Id for a packet.
- 2) The Time tells us the time at which the packet was captured. The first packets has always a time 0.000000 so if we see the time of last packets it will give us the time till which the capturing was done. We can also change the time display format by going to view and then navigating to Time Display format and select according to our own wish. Here we have options like second, Deci second, millisecond, microsecond, nanosecond.
- 3) Source tell that where the packet came from.
- 4) Destination tells us where the packet is going.
- 5) Info gives us a small summary of the packet.

We can also change their positions of columns by clicking on and dragging to the desired position. We can also hide these columns by right clicking and going to display column and here we can untick the column we want to hide.

If we want to see all packet details then we can right click in the packet details area and then click “Expand all”.

WHAT IS A PACKET?

It is a package of information that we want to send to a different location. Now let’s examine some packets. I have downloaded a sample http cap file and opened it in the wireshark.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	145.254.160.237	65.208.228.223	TCP	62	80->3372 [SYN] Seq=0 win=8760 Len=0 MSS=1460 SACK_PERM=1
2	0.911310	65.208.228.223	145.254.160.237	TCP	62	80->3372 [SYN, ACK] Seq=0 Ack=1 win=5840 Len=0 MSS=1380 SACK_PERM=1
3	0.911310	145.254.160.237	65.208.228.223	TCP	54	3372->80 [ACK] Seq=1 Ack=1 win=9660 Len=0
4	0.911310	145.254.160.237	65.208.228.223	HTTP	533	GET /download.html HTTP/1.1
5	1.472116	65.208.228.223	145.254.160.237	TCP	54	80->3372 [ACK] Seq=1 Ack=480 win=6432 Len=0
6	1.682419	65.208.228.223	145.254.160.237	TCP	1434	[TCP segment of a reassembled PDU]
7	1.812606	145.254.160.237	65.208.228.223	TCP	54	3372->80 [ACK] Seq=480 Ack=1381 win=9660 Len=0
8	1.812606	65.208.228.223	145.254.160.237	TCP	1434	[TCP segment of a reassembled PDU]
9	2.012894	145.254.160.237	65.208.228.223	TCP	54	3372->80 [ACK] Seq=480 Ack=2761 win=9660 Len=0
10	2.443513	65.208.228.223	145.254.160.237	TCP	1434	[TCP segment of a reassembled PDU]
11	2.553672	65.208.228.223	145.254.160.237	TCP	1434	[TCP segment of a reassembled PDU]
12	2.553672	145.254.160.237	65.208.228.223	TCP	54	3372->80 [ACK] Seq=480 Ack=5521 win=9660 Len=0

Frame 1: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)
 Ethernet II, Src: XeroxCor_00:00:00 (00:00:01:00:00:00), Dst: fe:ff:20:00:01:00 (fe:ff:20:00:01:00)

Now let's examine the packet 4 which is a http GET packet. So let's see its details

1)

```

Hypertext Transfer Protocol
  GET /download.html HTTP/1.1\r\n
  [Expert Info (Chat/Sequence): GET /download.html HTTP/1.1\r\n]
    Request Method: GET
    Request URI: /download.html
    Request Version: HTTP/1.1
    Host: www.ethereal.com\r\n
    User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.6) Gecko/20040113\r\n
    Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1\r\n
    Accept-Encoding: gzip,deflate\r\n
    Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
    Keep-Alive: 300\r\n
    Connection: keep-alive\r\n
    Referer: http://www.ethereal.com/development.html\r\n
  
```

So when
server

see this packet it thinks that ok it is some http request which means they are looking for some files or pictures or other web content then it checks the get method and which url is being requested. We also need to include basic tcp information in our packet.

```

Transmission Control Protocol, Src Port: 3372, Dst Port: 80, Seq: 1, Ack: 1, Len: 479
  Source Port: 3372
  Destination Port: 80
  [Stream index: 0]
  [TCP Segment Len: 479]
  Sequence number: 1 (relative sequence number)
  [Next sequence number: 480 (relative sequence number)]
  Acknowledgment number: 1 (relative ack number)
  Header Length: 20 bytes
  Flags: 0x018 (PSH, ACK)
    000. .... = Reserved: Not set
    ...0 .... = Nonce: Not set
    .... 0... = Congestion window reduced (CWR): Not set
    .... .0.. = ECN-Echo: Not set
    .... ..0. = Urgent: Not set
    .... ...1 = Acknowledgment: Set
    .... .... 1... = Push: Set
    .... .... .0.. = Reset: Not set
    0 - SYN: Not set
  
```

Like here in the above picture we can see the tcp information related to our packet like what ports we are using in our communication. We will also include ipv4 information since we are using ipv4 address.

```

Internet Protocol Version 4, Src: 145.254.160.237, Dst: 65.208.228.223
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    0000 00.. = Differentiated Services Codepoint: Default (0)
    .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
  Total Length: 519
  Identification: 0x0f45 (3909)
  Flags: 0x02 (Don't Fragment)
    0... .... = Reserved bit: Not set
    .1.. .... = Don't fragment: Set
    ..0. .... = More fragments: Not set
  Fragment offset: 0
  Time to live: 128
  Protocol: TCP (6)
  Header checksum: 0x9010 [validation disabled]
  [Header checksum status: Unverified]
  
```

And this holds the destination and the return address information. Now we will see the Ethernet information also as any data we send goes through some route. It is the information to connect our computer to the router and it uses the mac addressing.

```

Ethernet II, Src: XeroxCor_00:00:00 (00:00:01:00:00:00), Dst: fe:ff:20:00:01:00 (fe:ff:20:00:01:00)
  Destination: fe:ff:20:00:01:00 (fe:ff:20:00:01:00)
    Address: fe:ff:20:00:01:00 (fe:ff:20:00:01:00)
      ....1. .... = LG bit: Locally administered address (this is NOT the factory default)
      ....0. .... = IG bit: Individual address (unicast)
  Source: XeroxCor_00:00:00 (00:00:01:00:00:00)
    Address: XeroxCor_00:00:00 (00:00:01:00:00:00)
      ....0. .... = LG bit: Globally unique address (factory default)
      ....0. .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)
```

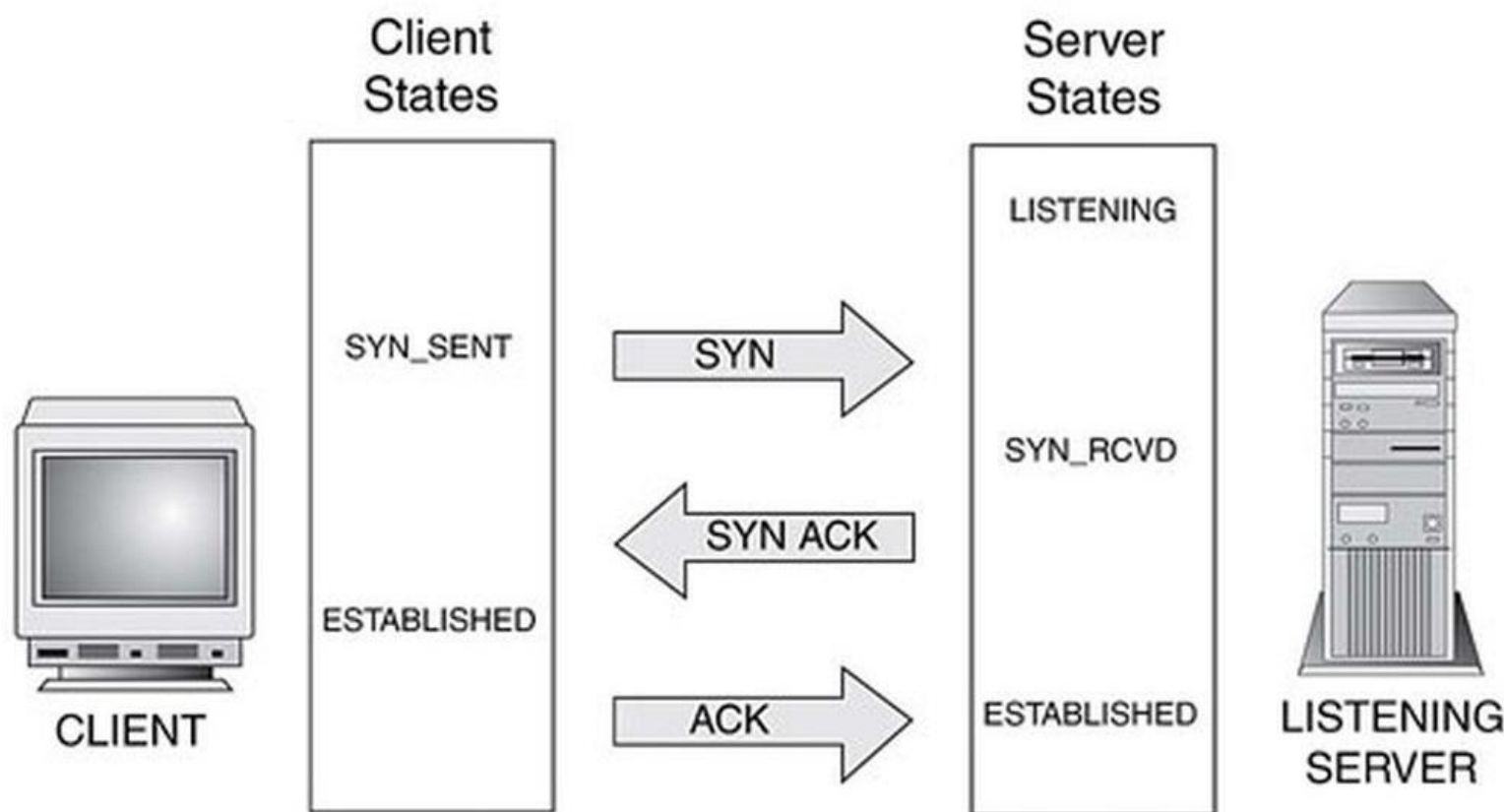
Now lets see what A frame is . A frame is a point where our softwares becomes real so here all these ones and zeros need to be converted into some signal so that it can get transmitted. Eg if we are sending data in Ethernet then it will generate electric pulses according to the ones and the zeros , if we are using fibre optics then light pulses will get generated. Once all the electrical pulses reach to server then the server interprete it and analyze the request.

```

Frame 4: 533 bytes on wire (4264 bits), 533 bytes captured (4264 bits)
```

TCP 3 WAY HANDSHAKE

http says what kind of data we are sending and how it should be formatted and read and in that way a server can easily read a packet.
TCP says that "This is how we are going to connect" It is not related to actual data or what we are sending.



First we will send a SYN request which is Synchronization request it is a way of saying to server that can I connect to you?

If the server allows the connection a SYN ACK(Synchronization Acknowledgement) is generated which is like saying I acknowledge that you made a request and yes you can connect to me. Now last part of this handshake is that this clients sends one last ACK . After the handshake we are now connected to the server and yes we can now send a http request. Now lets see the handshake in the following picture.

1	0.000000	145.254.160.237	65.208.228.223	TCP	62	3372-80	[SYN] Seq=0 win=8760 Len=0 MSS=1460 SACK_PERM=1	3 way tcp handshake
2	0.911310	65.208.228.223	145.254.160.237	TCP	62	80-3372	[SYN, ACK] Seq=0 Ack=1 win=5840 Len=0 MSS=1380 SACK_PERM=1	
3	0.911310	145.254.160.237	65.208.228.223	TCP	54	3372-80	[ACK] Seq=1 Ack=1 win=9660 Len=0	
4	0.911310	145.254.160.237	65.208.228.223	HTTP	533		GET /download.html HTTP/1.1	
5	1.472116	65.208.228.223	145.254.160.237	TCP	54	80-3372	[ACK] Seq=1 Ack=480 win=6432 Len=0	
6	1.682410	65.208.228.223	145.254.160.237	TCP	1424		TCP segment of a reassembled packet	

STEP1: Sending SYN request.

```

Header Length: 28 bytes
Flags: 0x002 (SYN)
000. .... = Reserved: Not set
...0 .... = Nonce: Not set
.... 0... = Congestion window Reduced (CWR): Not set
.... .0.. = ECN-Echo: Not set
.... ..0. = Urgent: Not set
.... ...0 = Acknowledgment: Not set
.... .... = Push: Not set
.... ..0.. = Reset: Not set
...1. .... = Syn: Set
[Expert Info (Chat/Sequence): Connection establish request (SYN): server port 80]
.... .... = Fin: Not set

```

All Other flags set to 0 as here we are sending the SYN REQUEST

step 1 SYN FLAG SET TO 1

STEP2: Sending SYN ACK request

```

Flags: 0x012 (SYN, ACK)
000. .... = Reserved: Not set
...0 .... = Nonce: Not set
.... 0... = Congestion window Reduced (CWR): Not set
.... .0.. = ECN-Echo: Not set
.... ..0. = Urgent: Not set
...1. .... = Acknowledgment: Set
.... .... = Push: Not set
.... ..0.. = Reset: Not set
...1. .... = Syn: Set
[Expert Info (Chat/Sequence): Connection establish acknowledge (SYN+ACK): server port 80]

```

OTHER FLAGS SET TO 0

ACK SET TO 1

OTHER FLAGS SET TO 0

SYN SET TO 1

STEP 3: GENERATING ACK

```

Flags: 0x010 (ACK)
000. .... = Reserved: Not set
...0 .... = Nonce: Not set
.... 0... = Congestion window Reduced (CWR): Not set
.... .0.. = ECN-Echo: Not set
.... ..0. = Urgent: Not set
...1. .... = Acknowledgment: Set
.... .... = Push: Not set
.... ..0.. = Reset: Not set
.... ...0 = Syn: Not set
.... .... = Fin: Not set
[TCP Flags: .....A....]
window size value: 9660

```

ONLY ACK IS SET TO 1 REST ALL ARE SET TO 0

IF WE ARE NOT GETTING ANY SYN ACK PACKETS THEN THE SERVER MIGHT BE DOWN OR YOU MAY BE BLOCKED BY A FIREWALL. AT THE END OF CONNECTION WE NEED TO SAY GOOD BYE TO THE SERVER. SO TO TERMINATE WE SEND A FIN PACKET AND THE SERVER RESPONDS BY GENERATING TCP FIN ACK AND AGAIN THE CLIENT GENERATES A ACK FOR THE LAST TIME.

