

**Title: Parsing IP packets**

**Name: Dikshya Kafle**

**Stu No: 2018380039**

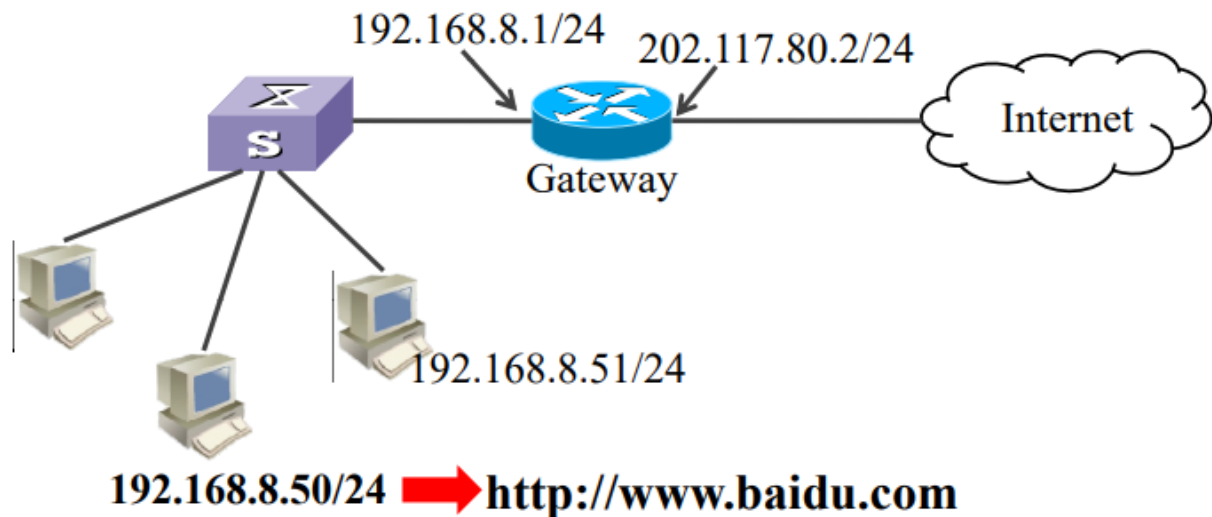
**Class No: 10101801**

**Deadline: Dec 13<sup>th</sup> 2020**

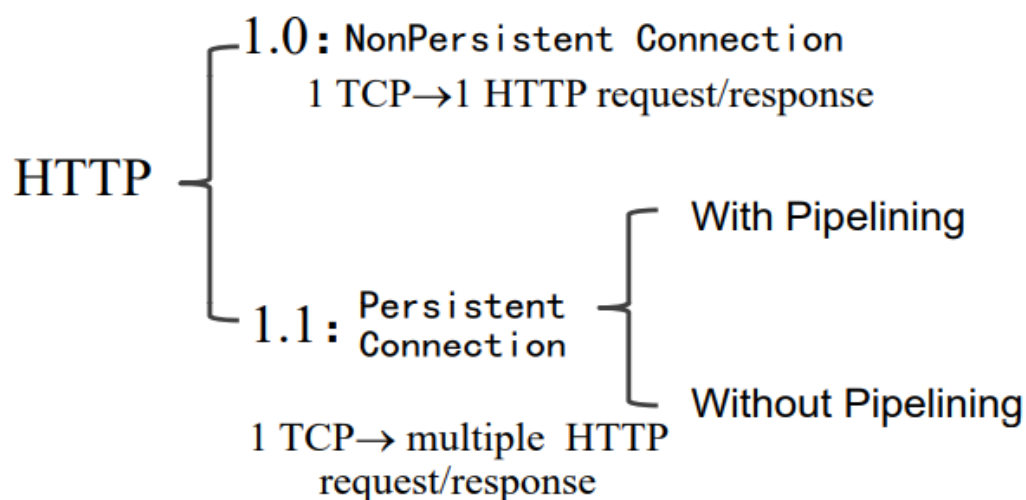
**School of Computer Science and Engineering**

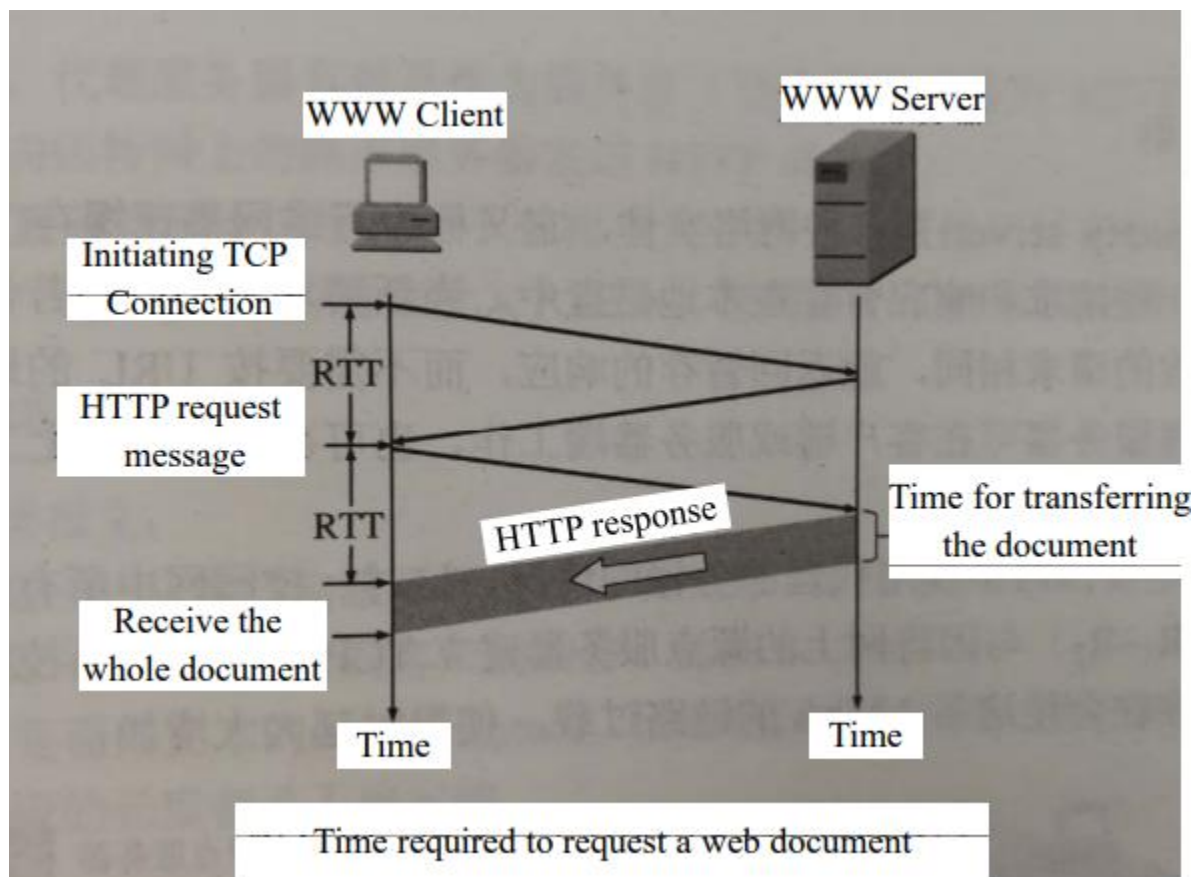
**Purpose:**

By analyzing IP packets, we can understand the working principle of IP, ARP, TCP, HTTP and DNS protocols

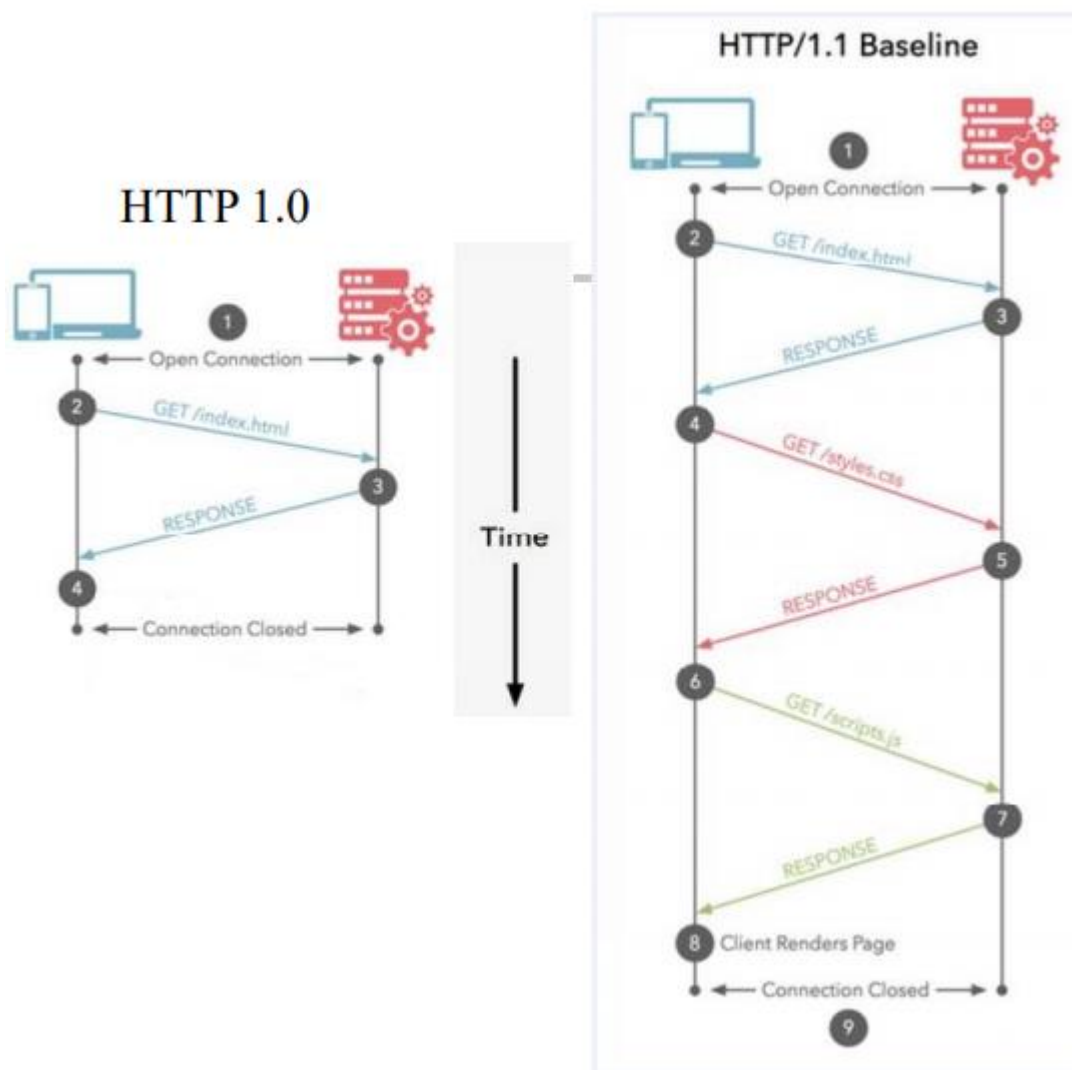
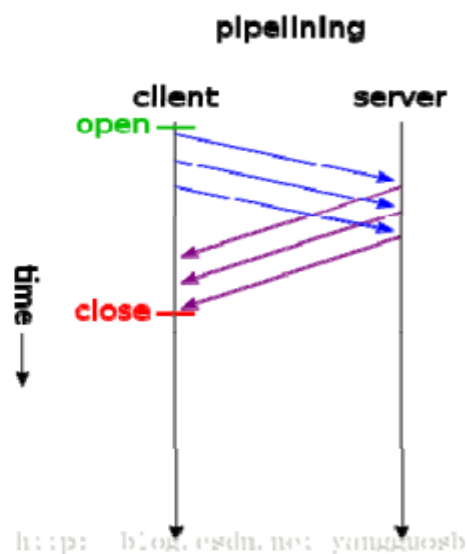
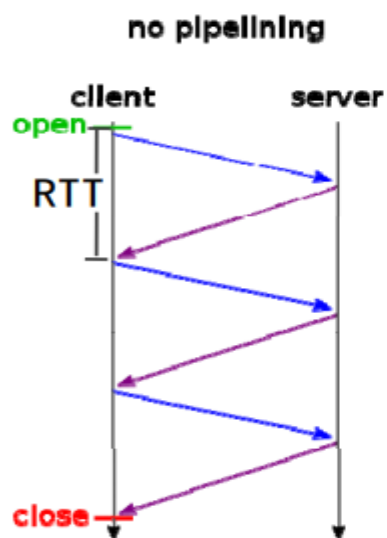
**Network Topology For Experiment:****Process of accessing webpage:**

1. Access <http://www.baidu.com>
2. Step 1. DNS UDP IP ARP recursion/iteration ICMP
3. Step 2. Establish TCP connection by three-way handshake.
4. Step 3. Client sends HTTP request.
5. Step 4. Server receive and return HTTP response.
6. Step 5. Release TCP connection by four-way wave hand.

**HTTP Working Mode:**



Pipeline Connection:

Pipeline Connection:

**Data encapsulation:**


DNS request	Frame header	IP Header	UDP Header	DNS Header	DATA	Frame Tail
-------------	--------------	-----------	------------	------------	------	------------


HTTP request	Frame header	IP Header	TCP Header	HTTP Header	Entity	Frame Tail
--------------	--------------	-----------	------------	-------------	--------	------------

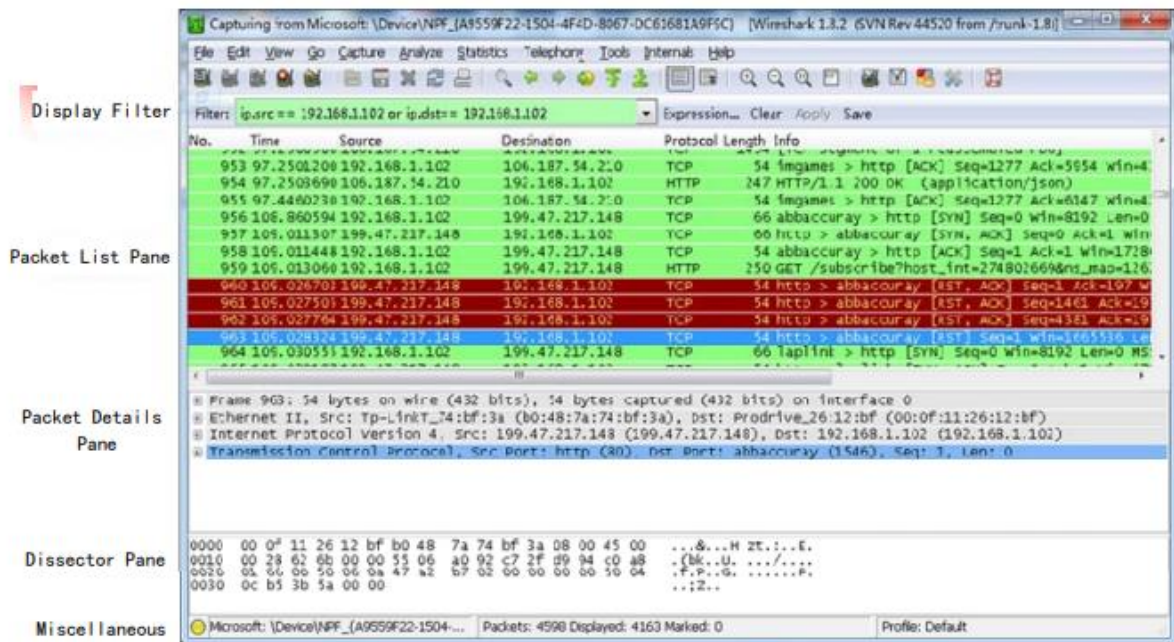
**ARP message format:****Byte**

2	Network type
2	Protocol type
1	Length for PA
1	Length for IP
2	Operation
6	Source PA
4	Source IP
6	Destination PA
4	Destination IP

**“1” denote Ether****“0x0800” is IP****PA: Physical address**

- 
 {
   
1—ARP request
   
2—ARP response
   
3—RARP request
   
4—RARP response


 Destination PA is empty in the request message



### 1- Preliminary:

(1) Clear browser cache Ensure that the Web is caught from network. Chrome: Options --> Under the Hood --> Clear browsing data.

### Clear browsing data

**Basic** Advanced

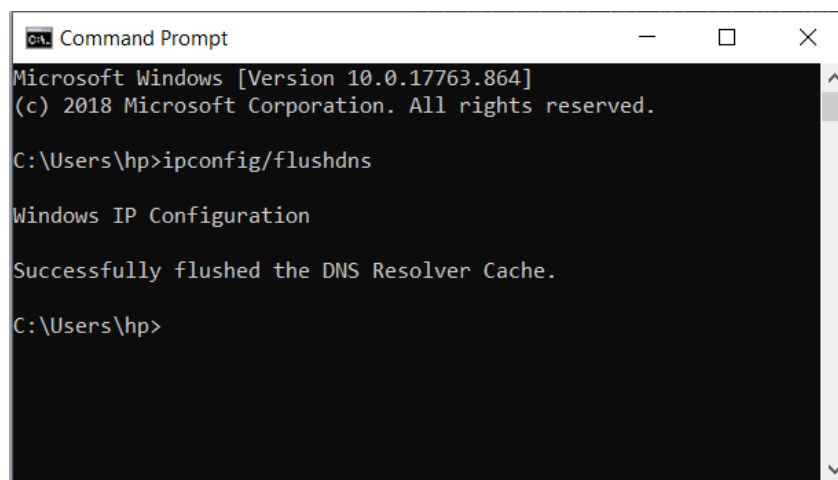
---

Time range: All time ▼

- ☒ **Browsing history**  
Clears history and autocompletions in the address bar.
- ☒ **Cookies and other site data**  
Signs you out of most sites.
- ☒ **Cached images and files**  
Frees up 200 MB. Some sites may load more slowly on your next visit.

Cancel Clear data

(2) Clear DNS cache Ensure that that the map of domain name and ip is got from network request. In Windows XP, input `ipconfig /flushdns`.



```
C:\> Command Prompt
Microsoft Windows [Version 10.0.17763.864]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\hp>ipconfig/flushdns

Windows IP Configuration

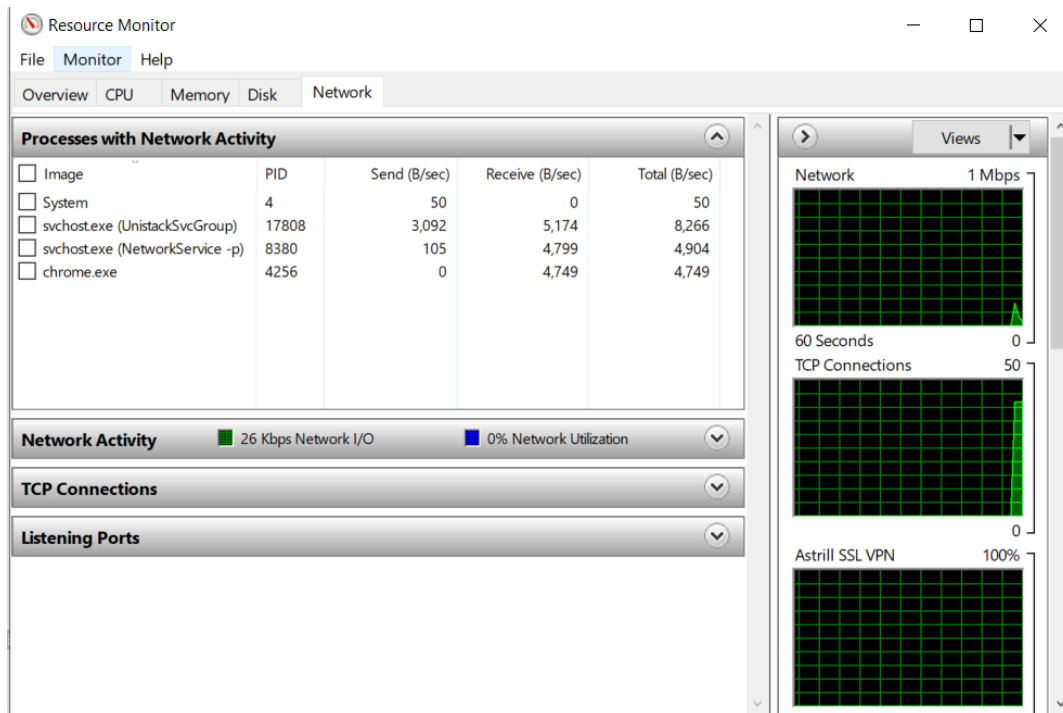
Successfully flushed the DNS Resolver Cache.

C:\Users\hp>
```

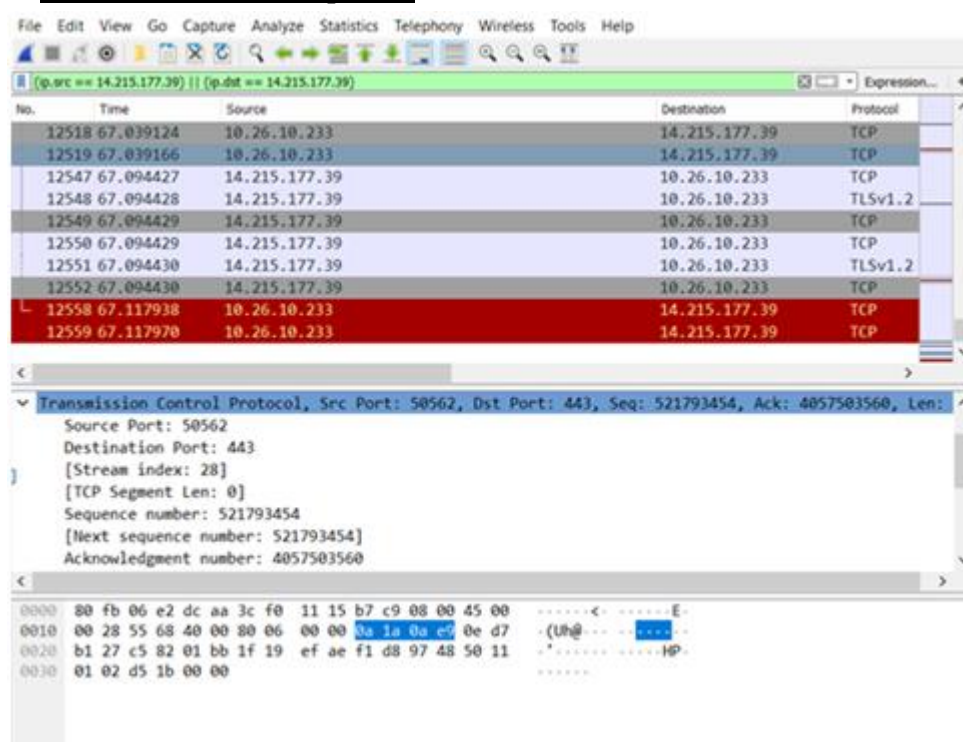
(3) Set filter rules In order to facilitate the analysis, set filter rules before catching the packets. In Filter ToolBar, Enter filter rule normal expression.

```
(ip.src == 14.215.177.39) || (ip.dst == 14.215.177.39)
```

#### (4) Close network applications



## 2- Start wireshark and Input url







### 3- Ip address of baidu.com

```
Command Prompt

baidu.com
-----
Record Name . . . . . : baidu.com
Record Type . . . . . : 1
Time To Live . . . . . : 510
Data Length . . . . . : 4
Section . . . . . : Answer
A (Host) Record . . . : 39.156.69.79

Record Name . . . . . : baidu.com
Record Type . . . . . : 1
Time To Live . . . . . : 510
Data Length . . . . . : 4
Section . . . . . : Answer
A (Host) Record . . . : 220.181.38.148

Record Name . . . . . : ns4.baidu.com
Record Type . . . . . : 1
Time To Live . . . . . : 510
Data Length . . . . . : 4
Section . . . . . : Additional
A (Host) Record . . . : 14.215.178.80

Record Name . . . . . : ns3.baidu.com
Record Type . . . . . : 1
Time To Live . . . . . : 510
```

Or we can see DNS query and response in wireshark:

Packet list	Narrow & Wide	Case sensitive	Display filter	Find	Cancel	
No.	Time	Source	Destination	Protocol	Length	Info
3624	26.626030	10.26.10.233	202.117.80.6	DNS	78	Standard query 0x4686 A auth.grammarly.com
3628	26.663990	10.26.10.233	202.117.80.6	DNS	80	Standard query 0xc808 PTR 45.200.58.216.in-addr.arpa
3632	26.668245	10.26.10.233	202.117.80.6	DNS	87	Standard query 0x1fe0 PTR 254.197.79.204.in-addr.arpa
3640	26.698527	10.26.10.233	202.117.80.6	DNS	84	Standard query 0xd457 A s2.googleusercontent.com
3641	26.701704	202.117.80.6	10.26.10.233	DNS	384	Standard query response 0xd457 A s2.googleusercontent.com CNAME googlehosted1.googleusercontent.c...
3655	26.728124	202.117.80.6	10.26.10.233	DNS	157	Standard query response 0x1fe0 No such name PTR 254.197.79.204.in-addr.arpa SOA ns1.msedge.net
3676	26.833541	202.117.80.6	10.26.10.233	DNS	439	Standard query response 0x4686 A auth.grammarly.com A 3.215.60.11 A 18.214.210.59 A 52.6.10.250 NS...
3795	27.661785	10.26.10.233	202.117.80.6	DNS	78	Standard query 0xa41c A data.grammarly.com
3804	27.772615	202.117.80.6	10.26.10.233	DNS	488	Standard query response 0xa41c A data.grammarly.com CNAME prod-data.acquisition.grammarlyaws.com A...
4068	29.943497	10.26.10.233	202.117.80.6	DNS	88	Standard query 0xafc5 A f-log-extension.grammarly.io
4083	30.057315	202.117.80.6	10.26.10.233	DNS	452	Standard query response 0xafc5 A f-log-extension.grammarly.io A 52.86.64.36 A 54.236.84.111 A 18.2...
4120	30.325855	10.26.10.233	202.117.80.6	DNS	60	Standard query 0xc70b A baidu.com
4130	30.330575	202.117.80.6	10.26.10.233	DNS	271	Standard query response 0xc70b A baidu.com A 39.156.69.79 A 220.181.38.148 NS ns3.baidu.com MS ns4...
4121	30.334348	10.26.10.233	202.117.80.6	DNS	73	Standard query 0x5700 A www.baidu.com
4122	30.339458	202.117.80.6	10.26.10.233	DNS	302	Standard query response 0x5700 A www.baidu.com CNAME www.a.shifen.com A 14.215.177.39 A 14.215.177...
4325	30.974367	10.26.10.233	202.117.80.6	DNS	75	Standard query 0xf961 A bl1dstatic.com
Internet Protocol Version 4, Src: 10.26.10.233, Dst: 202.117.80.6						
User Datagram Protocol, Src Port: 54846, Dst Port: 53						
Domain Name System (query)						
Transaction ID: 0x670b						
Flags: 0x0100 Standard query						
Questions: 1						
Answer RRs: 0						
Authority RRs: 0						
Additional RRs: 0						
Queries						
1 baidu.com type A, class IN [Response in: 4120]						

No.	Time	Source	Destination	Protocol	Length	Info
15450	133.824188	HuaweiTe_40:b8:0e	Broadcast	ARP	56	Who has 10.26.0.1? Tell 10.26.64.132
15453	133.864621	Oneplus_92:ec:46	Broadcast	ARP	56	Who has 10.26.91.135? Tell 10.26.91.223
15456	133.881865	SichuanA_44:52:77	Broadcast	ARP	56	Who has 10.26.50.237? Tell 10.26.137.122
15462	133.979621	SichuanA_44:52:77	Broadcast	ARP	56	Who has 10.26.168.56? Tell 10.26.137.122
15466	134.026773	HuaweiTe_84:64:bb	Broadcast	ARP	56	Who has 10.26.0.1? Tell 10.26.44.66
15467	134.045195	RealmeCh_f7:88:a5	Broadcast	ARP	56	Who has 10.26.0.1? Tell 10.26.142.176
15472	134.066722	Oneplus_92:ec:46	Broadcast	ARP	56	Who has 10.26.91.105? Tell 10.26.91.223
15478	134.146676	RealmeCh_f7:88:a5	Broadcast	ARP	56	Who has 10.26.0.1? Tell 10.26.142.176
15479	134.161643	IntelCor_15:b7:c9	HuaweiTe_e2:dc:aa	ARP	42	Who has 10.26.0.1? Tell 10.26.10.233
15480	134.165270	HuaweiTe_e2:dc:aa	IntelCor_15:b7:c9	ARP	56	10.26.0.1 is at 80:fb:06:e2:dc:aa
15481	134.169259	HuaweiTe_d1:32:64	Broadcast	ARP	56	Who has 10.26.122.222? Tell 10.26.122.233
15485	134.211817	SichuanA_44:52:77	Broadcast	ARP	56	Who has 10.26.142.203? Tell 10.26.137.122
15486	134.221689	SichuanA_44:52:77	Broadcast	ARP	56	Who has 10.26.101.11? Tell 10.26.137.122
15487	134.239160	Azureway_79:91:3d	Broadcast	ARP	56	Who has 169.254.186.234? Tell 10.26.253.187
15488	134.244631	RealmeCh_f7:88:a5	Broadcast	ARP	56	Who has 10.26.0.1? Tell 10.26.142.176
15504	134.355101	Oneplus_92:ec:46	Broadcast	ARP	56	Who has 10.26.91.135? Tell 10.26.91.223
15508	134.555383	SichuanA_44:52:77	Broadcast	ARP	56	Who has 10.26.130.103? Tell 10.26.137.122

Type: ARP (0x0806)  
Trailer: 00000000000000000000000000000000

Address Resolution Protocol (reply)  
Hardware type: Ethernet (1)  
Protocol type: IPv4 (0x0800)  
Hardware size: 6  
Protocol size: 4  
Opcode: reply (2)  
Sender MAC address: HuaweiTe\_e2:dc:aa (80:fb:06:e2:dc:aa)  
Sender IP address: 10.26.0.1  
Target MAC address: IntelCor\_15:b7:c9 (3c:f0:11:b5:b7:c9)  
Target IP address: 10.26.10.233

```

0000 3c f0 11 15 b7 c9 80 fb 06 e2 dc aa 08 06 00 01 <----->
0010 08 00 06 04 00 02 80 fb 06 e2 dc aa 0a 1a 00 01
0020 3c f0 11 15 b7 c9 0a 1a 0a e9 00 00 00 00 00 00 <----->
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 <----->

```

## 5- ISN of data? Server ISN

1107 3.844464	10.26.10.233	14.215.177.39	TCP	66 50561 → 443 [SYN] Seq=1125593157 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
1108 3.844541	10.26.10.233	14.215.177.39	TCP	66 50562 → 443 [SYN] Seq=521785216 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
1116 3.883372	14.215.177.39	10.26.10.233	TCP	66 443 → 50561 [SYN, ACK] Seq=1713085514 Ack=1125593157 Win=8192 Len=0 MSS=1448 WS=32 SACK_PERM=1
1117 3.883372	14.215.177.39	10.26.10.233	TCP	66 443 → 50562 [SYN, ACK] Seq=4057327435 Ack=521785217 Win=8192 Len=0 MSS=1448 WS=32 SACK_PERM=1

```

> Frame 1107: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
> Ethernet II, Src: IntelCor_15:b7:c9 (3c:f0:11:15:b7:c9), Dst: HuaweiTe_e2:dc:aa (80:fb:06:e2:dc:aa)
> Internet Protocol Version 4, Src: 10.26.10.233, Dst: 14.215.177.39
▼ Transmission Control Protocol, Src Port: 50561, Dst Port: 443, Seq: 1125593157, Len: 0
    Source Port: 50561
    Destination Port: 443
    [Stream index: 27]
    [TCP Segment Len: 0]
    Sequence number: 1125593157
    [Next sequence number: 1125593157]
    Acknowledgment number: 0
    1000 .... = Header Length: 32 bytes (8)

```

The client ISN is: 1125593157

The server ISN is: 1713085514

Note that the browser will set multiple TCP connection at the same time with the server to make a parallel retrieve of information as only one document can be asked per TCP connection, that's why we find multiple SYN requests.

## 6- HTTP version, working mode

The screenshot shows a Wireshark capture of an HTTP GET request and its corresponding 304 Not Modified response. The packet list shows the request from 10.27.198.135 to 113.133.46.1. The packet details pane shows the response structure: HTTP/1.1 304 Not Modified. The packet bytes pane shows the raw data of the response, including the status line and headers.

## 7- One TCP connection, Amount of data sent? Amount of data received

Protocol	Length	Info
TCP	66	50561 → 443 [SYN] Seq=1125593157 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
TCP	66	50562 → 443 [SYN] Seq=521785216 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
TCP	66	443 → 50561 [SYN, ACK] Seq=1713085514 Ack=1125593158 Win=8192 Len=0 MSS=1448 WS=32 SACK_PERM=1
TCP	66	443 → 50562 [SYN, ACK] Seq=4057327435 Ack=521785217 Win=8192 Len=0 MSS=1448 WS=32 SACK_PERM=1
TCP	54	50561 → 443 [ACK] Seq=1125593158 Ack=1713085515 Win=66560 Len=0
TCP	54	50562 → 443 [ACK] Seq=521785217 Ack=4057327436 Win=66560 Len=0

We can see in these two TCP packets, the length of data is clearly 0 as the client is only trying to establish the connection so it sends no data and the server only acknowledges the request and sends no data two. That why len=0.

However we can see another TCP packet where the payload length is 1440 bytes:

```

> Flags: 0x010 (ACK)
  Window size value: 944
  [Calculated window size: 30208]
  [Window size scaling factor: 32]
  Checksum: 0x0c39 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
> [SEQ/ACK analysis]
> [Timestamps]
  TCP payload (1440 bytes)
  [Reassembled PDU in frame: 1142]
  TCP segment data (1440 bytes)

```

0020	0a e9 01 bb c5 82 f1 d5 e7 90 1f 19 d1 86 50 10	.....P.
0030	03 b0 0c 39 00 00 16 03 03 0e 2d 0b 00 0e 29 00	...9... ..)
0040	0e 26 00 09 b3 30 82 09 af 30 82 08 97 a0 03 02	..&...0.. -0.....
0050	01 02 02 0c 2c ee 19 3c 18 82 78 ea 3e 43 75 73	....,.< ..x.>Cus
0060	30 0d 06 09 2a 86 48 86 f7 0d 01 01 0b 05 00 30	0...*.H. ....0
0070	66 31 0b 30 09 06 03 55 04 06 13 02 42 45 31 19	f1.0...U ....BE1.
0080	30 17 06 03 55 04 0a 13 10 47 6c 6f 62 61 6c 53	0...U... -GlobalS
0090	69 67 6e 20 6e 76 2d 73 61 31 3c 30 3a 06 03 55	ign nv-s a1<0:...U
00a0	04 03 13 33 47 6c 6f 62 61 6c 53 69 67 6e 20 4f	...3Glob alSign 0
00b0	72 67 61 6e 69 7a 61 74 69 6f 6e 20 56 61 6c 69	rganizat ion Vali

Transmission Control Protocol (tcp), 20 bytes

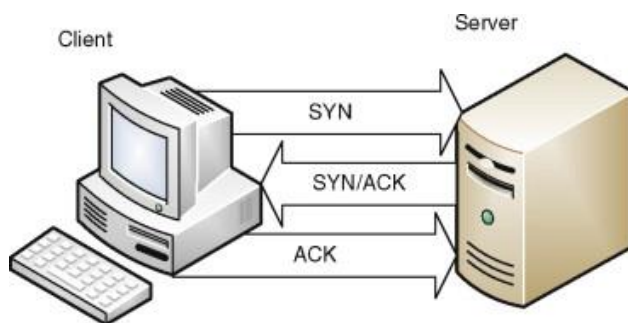
## 8- The process of three-way handshake connection and four-way wavehand release.

### Three-way handshake:

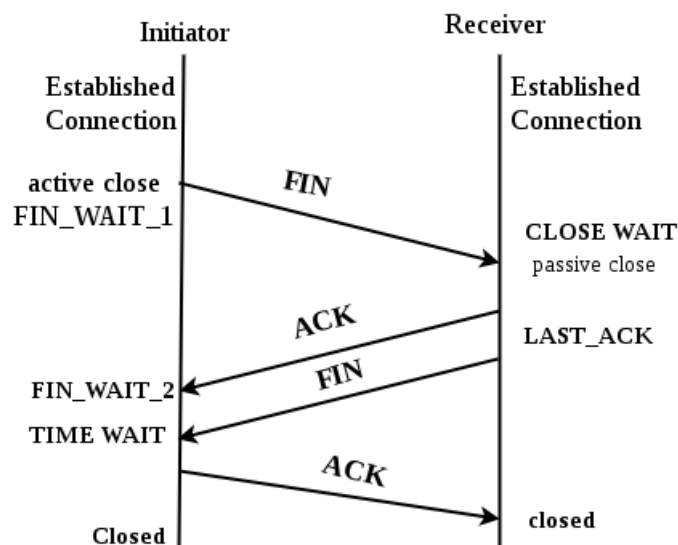
**Step 1 (SYN) :** In the first step, the client wants to create a server connection, so it sends a SYN (Synchronize Sequence Number) segment that tells the server that the client is likely to initiate contact and with which sequence number it begins segments with

**Step 2 (SYN + ACK):** The server responds with SYN-ACK signal bits set to a client request. Acknowledgement (ACK) refers to the reaction of the segment it obtained and SYN indicates with what sequence number the segments are likely to start with

**Step 3 (ACK):** The client acknowledges the server's response in the final part and both create a secure link with which they start the real data transfer.



#### Four-way wavehand release.



**Step 1 (FIN From Client)** – Suppose that the client application decides it wants to close the connection. (Note that the server could also choose to close the connection). This causes the client send a TCP segment with the FIN bit set to 1 to server and waits for a TCP segment from the server with an acknowledgment (ACK).

**Step 2 (ACK From Server)** – When Server received FIN bit segment from Sender (Client), Server Immediately send acknowledgement (ACK) segment to the Sender (Client).

**Step 3 (Client waiting)** – The client then waits for a TCP segment from the server with an acknowledgment. When it receives this segment, the client waits for another segment from the server with the FIN bit set to 1.

**Step 4 (FIN from Server)** – Server sends FIN bit segment to the Sender(Client) after some time when Server send the ACK segment.

**Step 5 (ACK from Client)** – When Client receive FIN bit segment from the Server, the client acknowledges the server's segment.

**Summary:**

We have finished running the software and produced a program that produces the values of the IP packet field and the data together. We will now explain the details of each byte that we use in the TCP header and UDP header packets. We have also examined IP packets and understood the working principle of the IP protocol.