Network Fundamental

DevOps Academy Project

DAY 1

Instructor

Praparn L.

Email: eva10409@gmail.com
Github: https://github.com/praparn
Line: praparn.l

Course Description

- Duration: 2 Days (09.00 - 17.30)
- Activity ratio:
  - Theory / Lecture (40%)
  - Practice / Workshop / Discussion (60%)
- Brake period:
  - 12.00 - 13.00 (Lunch)
- Facility:
  - Notebook for workshop (BYOD) (Windows, MacOSX)
  - Internet access
  - Projector
  - Lab resource (provide by instructor)

Workshop Resource:

- All workshop running on cloud with ubuntu 18.09
- Attendee need to ssh connection from your computer to cloud for access workshop
  - Windows: putty, wsl with terminal
  - Mac: terminal
  - Linux: terminal
- SSH-Key are available on thumb drive
- Instruction for ssh to cloud will available on workshop folder

Agenda

- **Day1**
- Network and Communication Concept
  - Concept of communication
  - Protocol on referencing
  - Conductor/Carrier
- Walkthrough OSI 7 Layer
  - What is OSI module
  - Dataflow and encapsulation concept
  - L1 - L7 description
  - Advance concept:
    - VLAN (RFC 5517/IEEE802.1Q)
    - VXLAN (RFC7348)
- TCP/IP Architecture:
  - Introduction to TCP/IP
  - Reflect architecture between TCP/IP and OSI

Agenda

- **Day1 (Continue)**
- MAC Address operation (OSI: L2, TCP/IP: L1-2)
  - ARP operation
- IP Address and routing (OSI: L3, TCP/IP: L2)
  - Mathematic and IP address overview
  - IP Address structure, class (A,B,C) and CIDR
  - IP Private Addresses
  - NAT operation
  - Binary math for subnet mask
  - IP routing operation
  - ARP operation
  - IPv4 vs IPv6

Agenda

- **Day1 (Continue)**
- TCP/UDP flow control (OSI: L4, TCP/IP: L3)
  - Connection oriented vs Connectionless
  - TCP flow operate
    - TCP flow control and congestion control
      - TCP three way handshake and TCP terminate
      - Sliding windows
      - Timeout and Retransmission
      - TCP keep alive
  - UDP flow operate

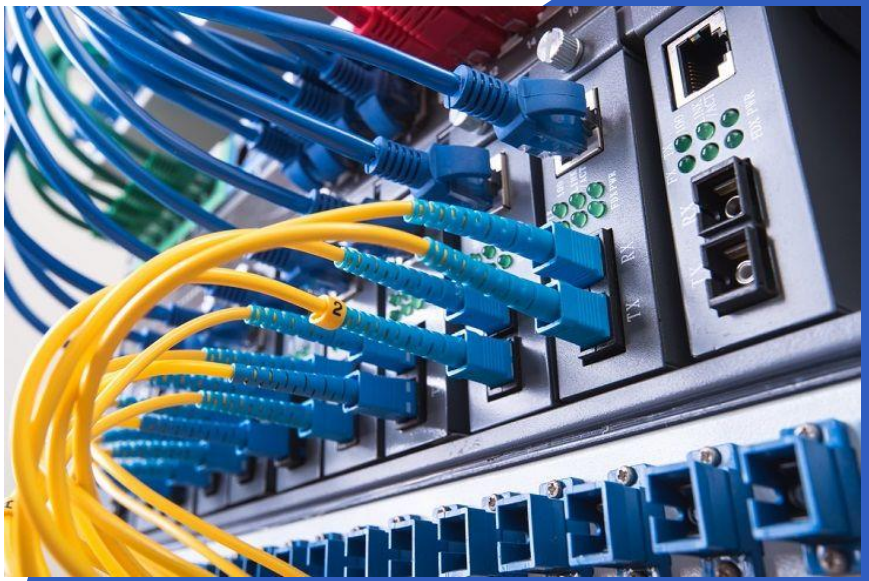
Agenda

- **Day2 (Continue)**
- Application layer flow control (OSI:L5 - L7, TCP/IP: L4)
  - DNS protocol (OSI: L6, TCP/IP: L4)
  - HTTP protocol (OSI: L7, TCP/IP: L4)
    - HTTP standard response code
    - HTTP generation
      - HTTP/0.9 - 1.1 (RFC:2616)
      - SPDY 2009, HTTP/2 (RFC:7540)
      - HTTP/3, QUIC (RFC:On review)
  - SSL/TLS protocol (OSI: L6+, TCP/IP: L4+)
      - SSL1.0, 2.0, 3.0 (RFC:6176, RFC:7568)
      - TLS1.0 (RFC: 2246)
      - TLS1.1 (RFC: 4346)
      - TLS1.2 (RFC: 5246)
      - TLS1.3 (RFC: 8446)

Agenda

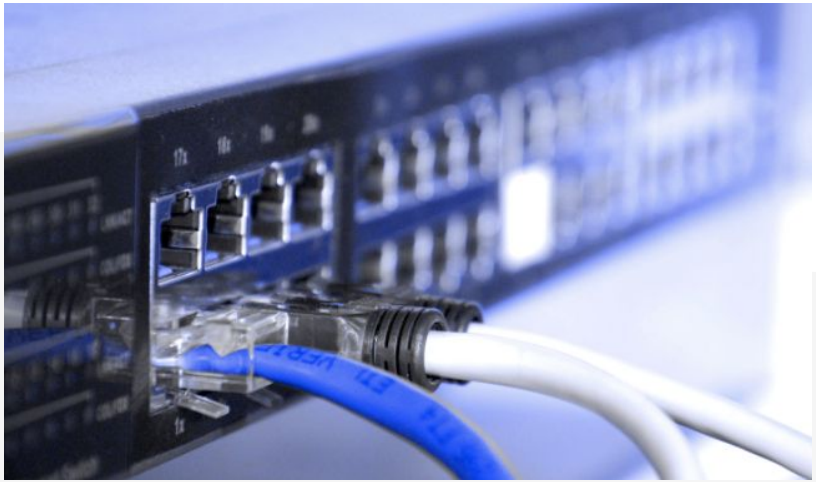
- **Day2 (Continue)**
- Application layer flow control (OSI:L5 - L7, TCP/IP: L4)
  - Websocket and HTTP
- Advance network operation
  - ICMP operation
  - Linux IPTables operation
  - Linux IPVS load balance

Network and Communication Concept



“The network is the computer”

—SCOTT MCNEALY

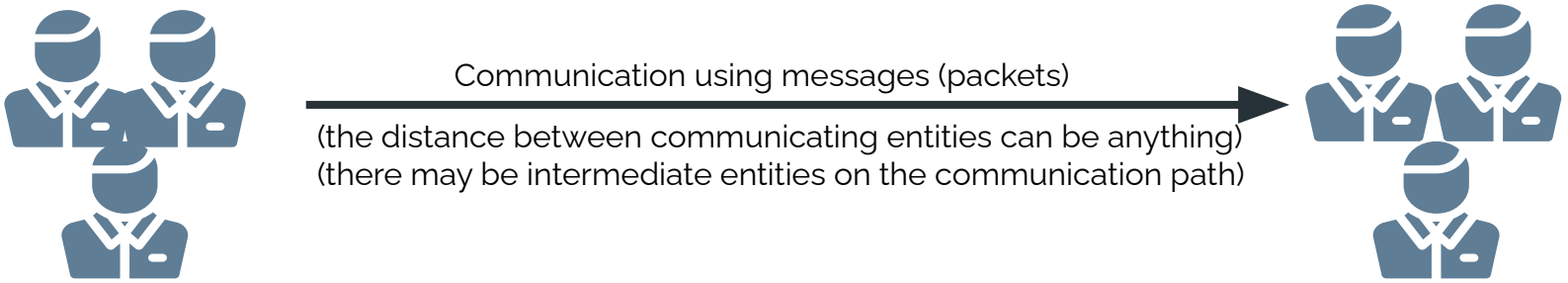


Concept of Communication

- **Communication** “is the act of conveying meanings from one entity or group to another through the use of mutually understood signs, symbols, and semiotic rules”. - Wikipedia
- **Computer network** “is a group of computers that use a set of common communication protocols over digital interconnections for the purpose of sharing resources located on or provided by the network nodes” - Wikipedia
- **Protocols:**
  - Common language for communication
  - Must be standard for all party (vendor) reference and follow when building networking equipment and software

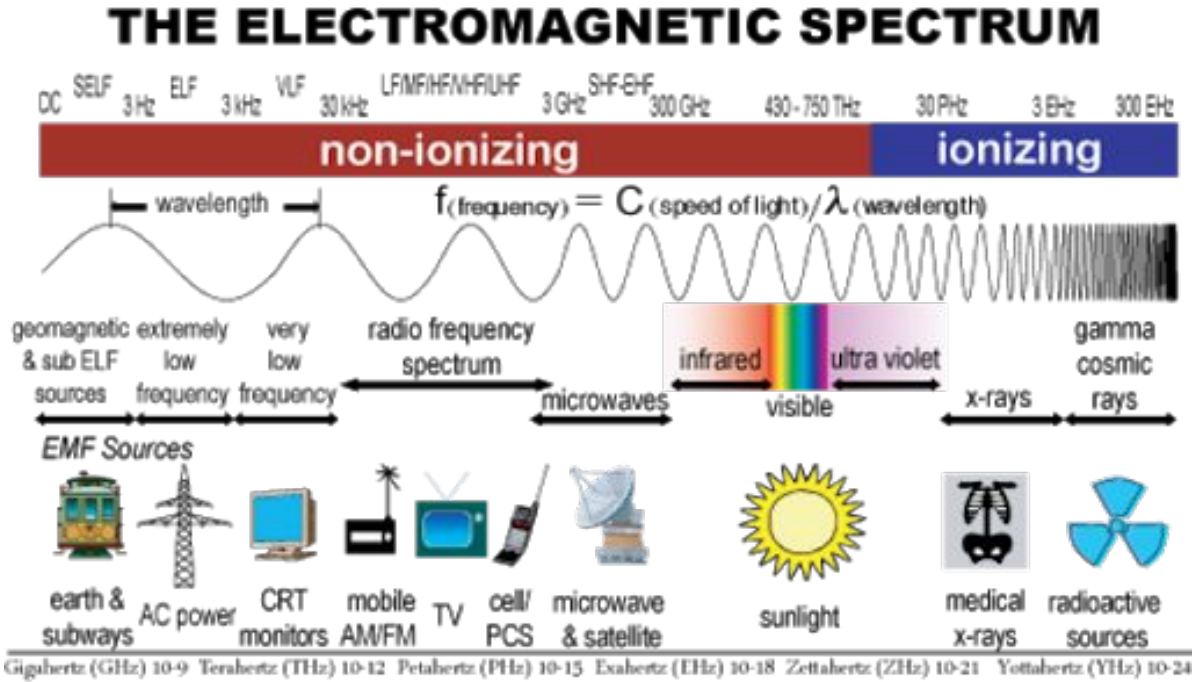
Protocol for Reference

- **A protocol defines** “types”, “formats” and “order” of messages (packets) exchanged between communication entities
- **Protocol** have role to defines the actions that the entity should perform on receipt/transmission of these message

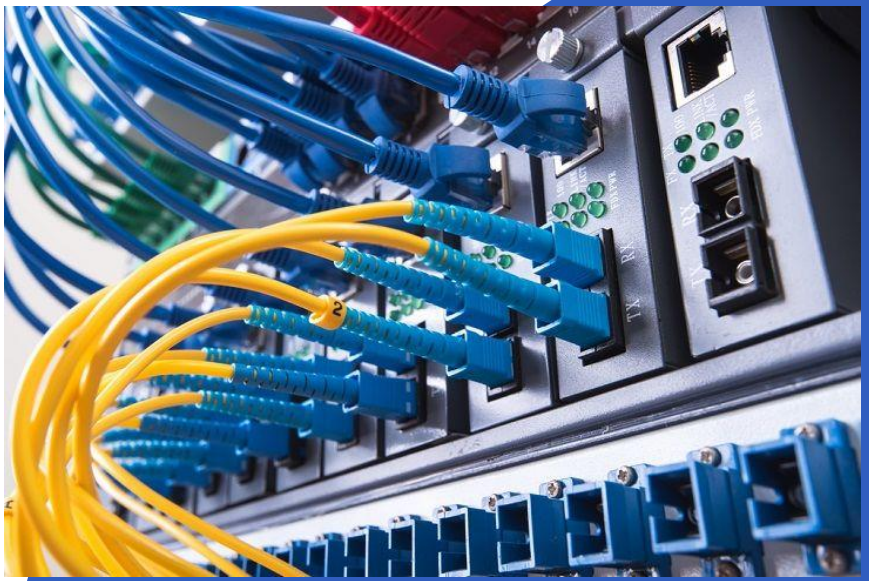


Conductor/Carrier

- Network communication need “Conductor/Carrier”
  - Electric (Ex: copper, metal, silver etc)
  - Electromagnetic (Ex: microwave, satellite, fm/am, wireless etc)



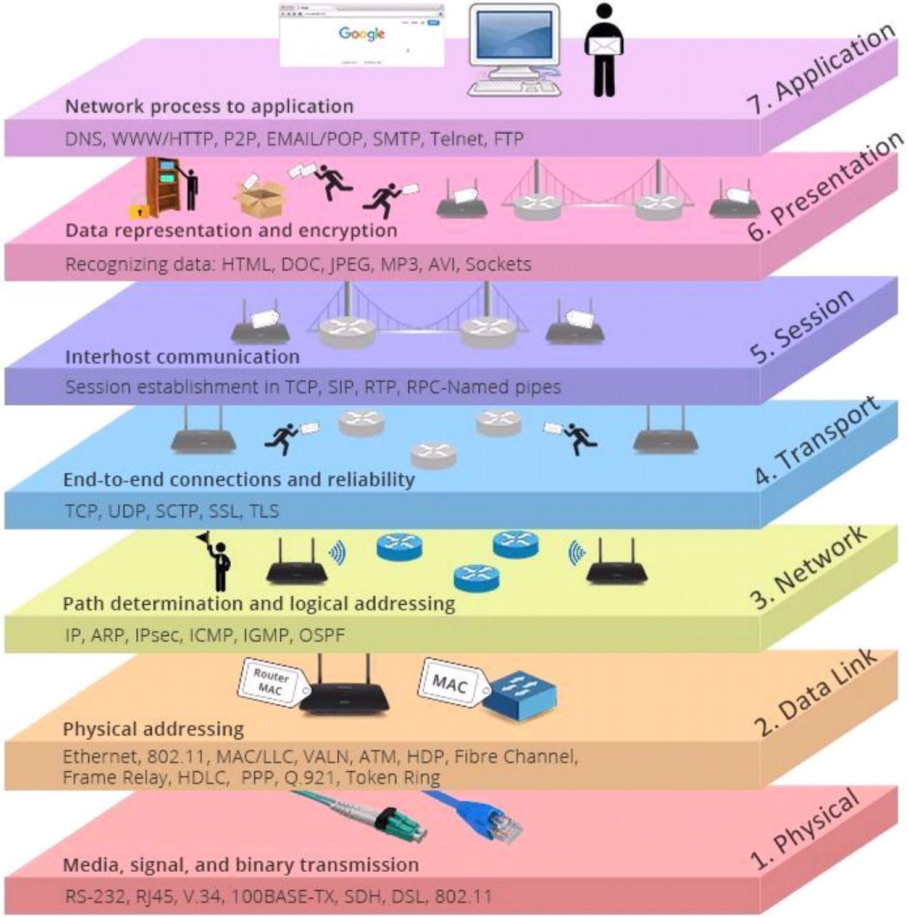
Walkthrough OSI 7 Layer Model



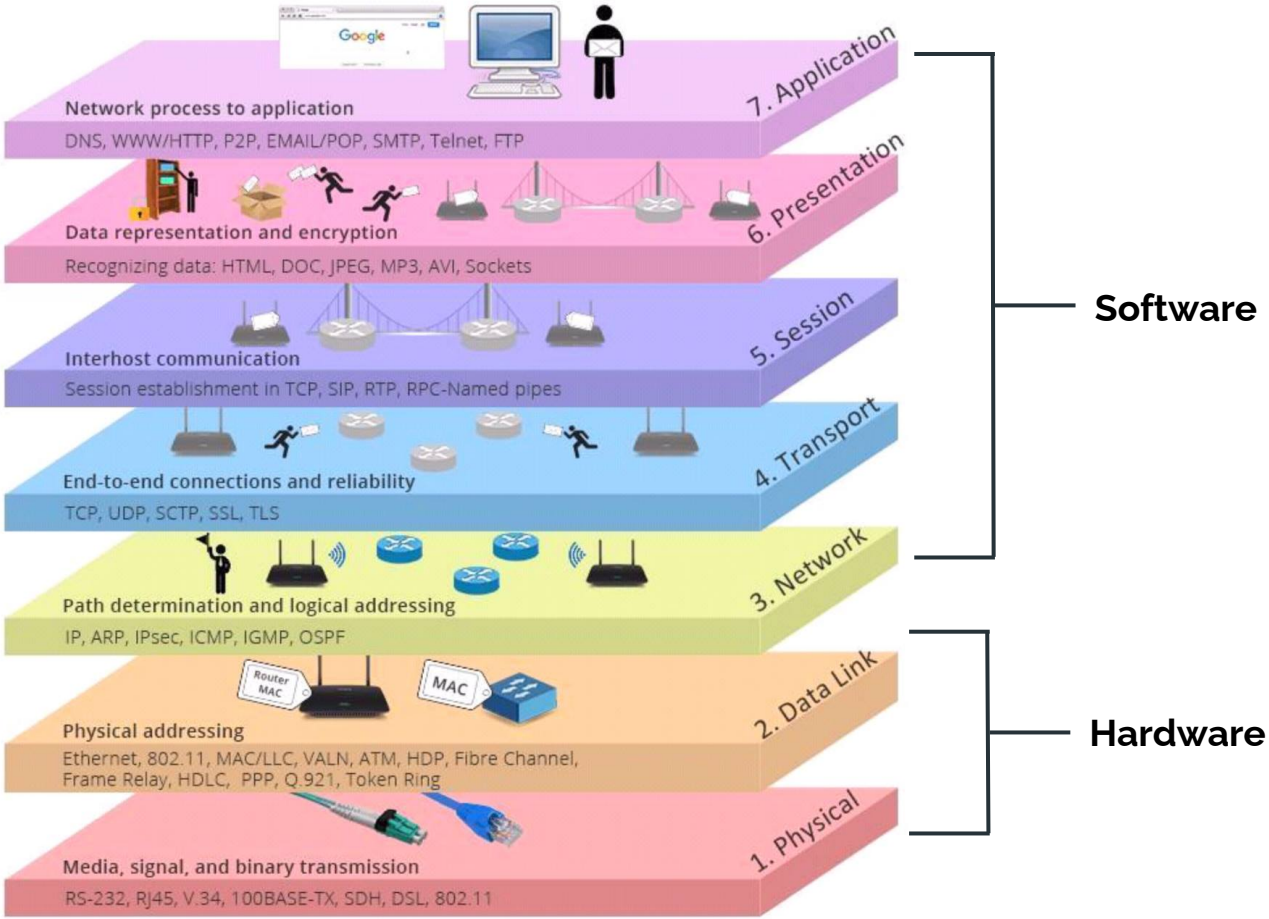
What is OSI Model

- OSI model was developed starting in the late 1970s and model became a working product at International Organization for Standardization (ISO) in 1980
- OSI model are defined role and duty for each layer of network communication. All vendor and manufacturer of network equipment need to reference OSI as standard for join computer network communication
- As purpose for design this model. OSI was designed as abstract and formal written in ISO/IEC 7498
    - ISO/IEC 7498-1 The Basic Model
    - ISO/IEC 7498-2 Security Architecture
    - ISO/IEC 7498-3 Naming and addressing
    - ISO/IEC 7498-4 Management framework

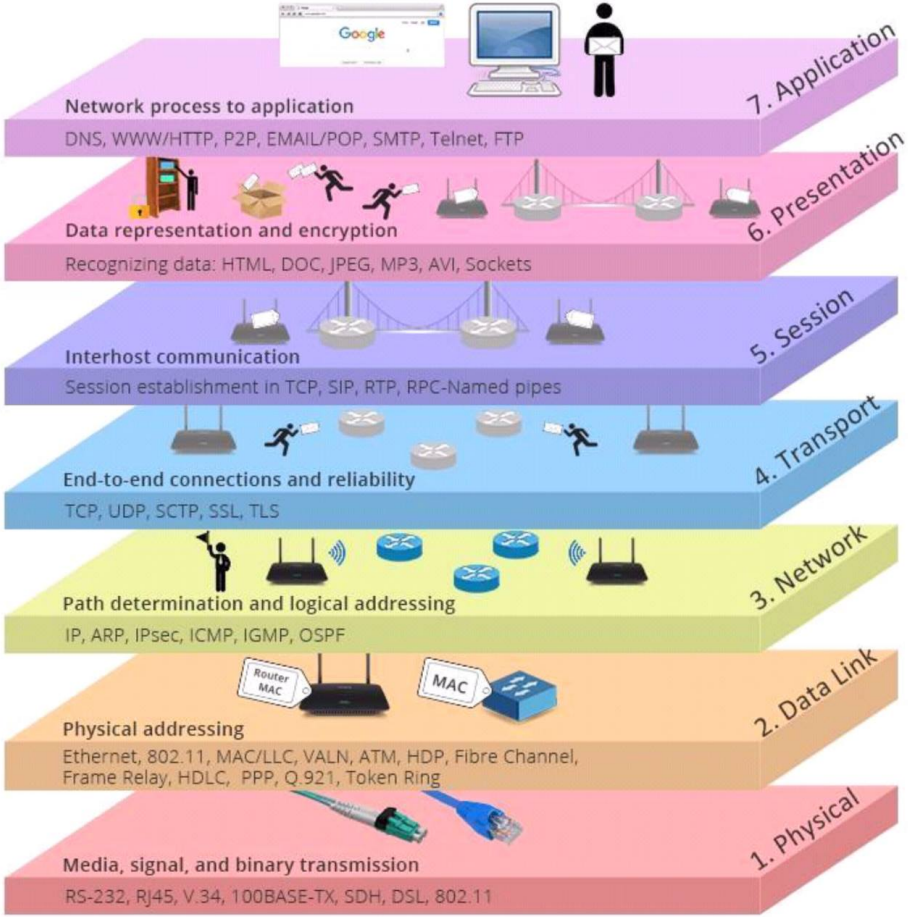
What is OSI Model



What is OSI Model (Hardware/Software)



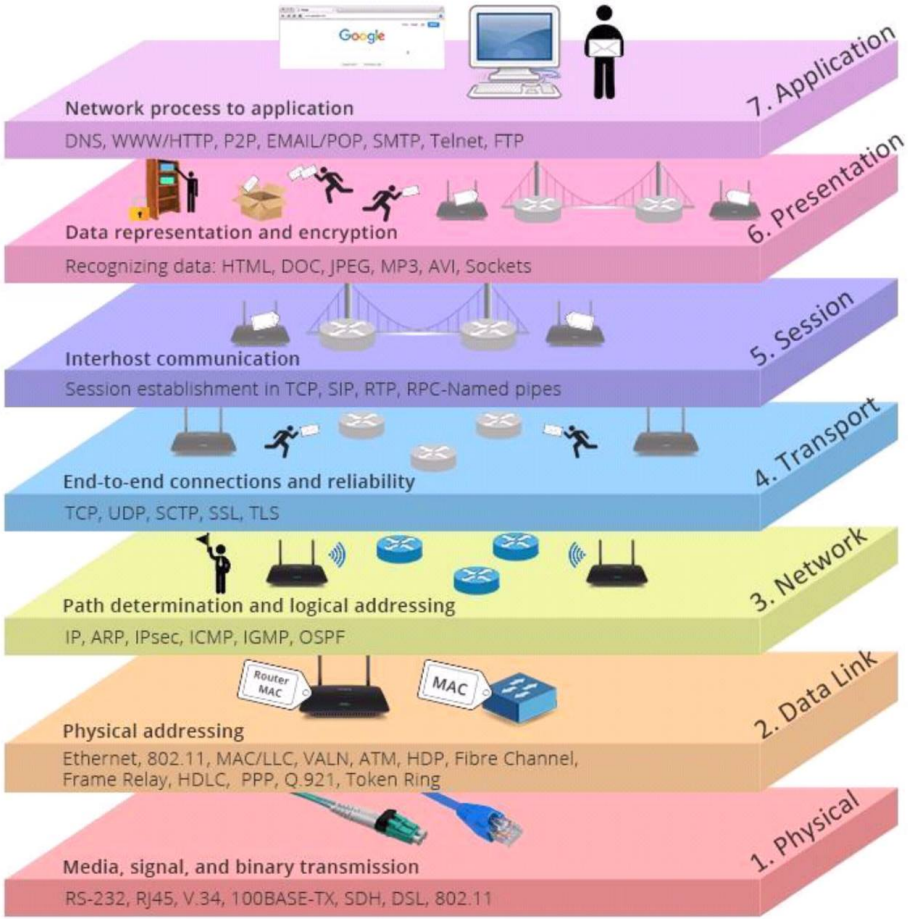
What is OSI Model (OS/Application)



Application Layer

OS Layer (DataFlow)

What is OSI Model



——Data

——Data

——Data

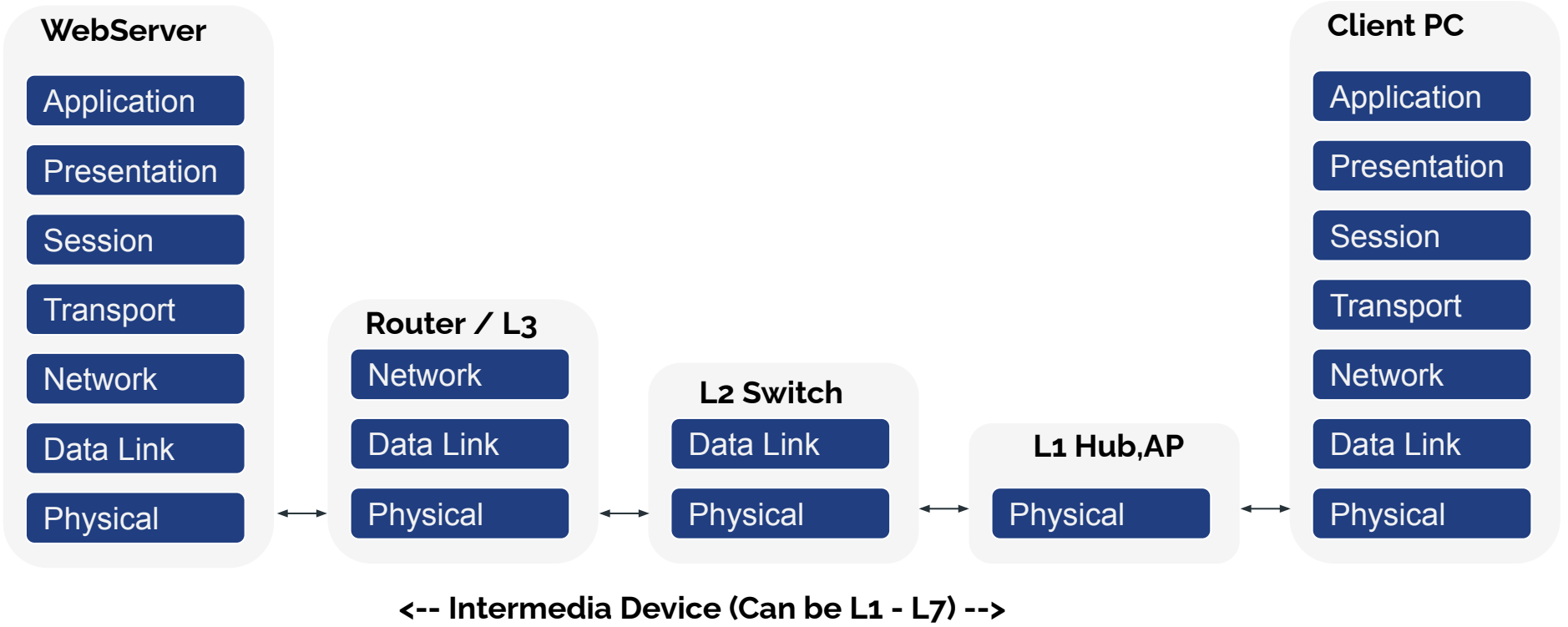
——Segments

——Packets (Router/L3 Switch)

——Frame (L2 Switch, Bridge, AP)

——Bits (Hub, Repeater, AP)

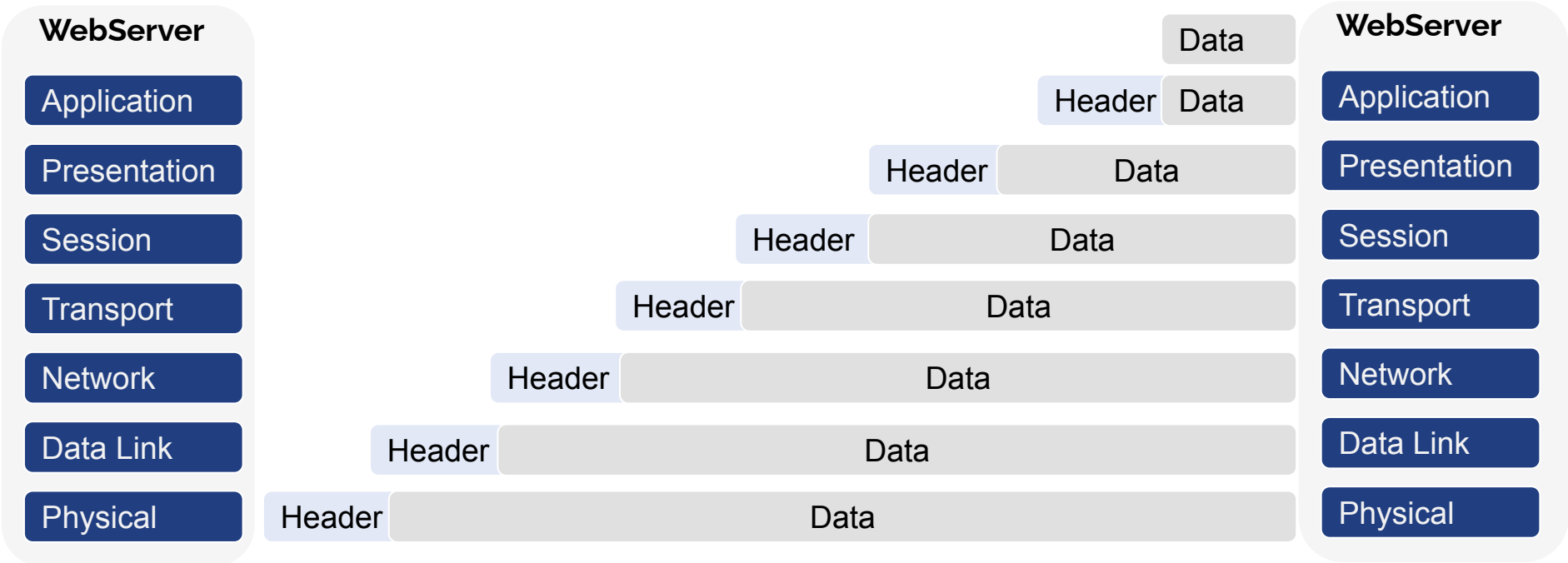
What is OSI Model



Dataflow and Encapsulation

- OSI will send data from upper layer to lower layer and control data on each layer by add “header” on packet.
- The lower layer will acknowledge entire packet from upper layer as “data” and add their “header” before send to lower layer again.
- This process will operate until packet reach “physical layer” and send out to destination
- We call this process as “Encapsulation”
- From this model each layer of OSI can communicate with peer on same layer similarly direct connect

OSI Model Encapsulation



Advance concept (VLAN)

- VLAN (Virtual LAN) (IEEE:802.1Q)
- Software define network since 1981 until now !!!
- VLAN is basic concept for create multiple LANs network for communication with same carrier with original purpose for segmentation large network to many small subnet and reduce broadcast/multicast traffic on network
- This technique is well-know for increase network flexible for user/host, bandwidth allocation and resource optimization
- IEEE 802.1Q was defined for operate standard for VLAN tagging
- VLAN is operate on L2 layer of OSI model by add 4 byte of ethernet frame after source address for VLAN tagging up to 4096 networks

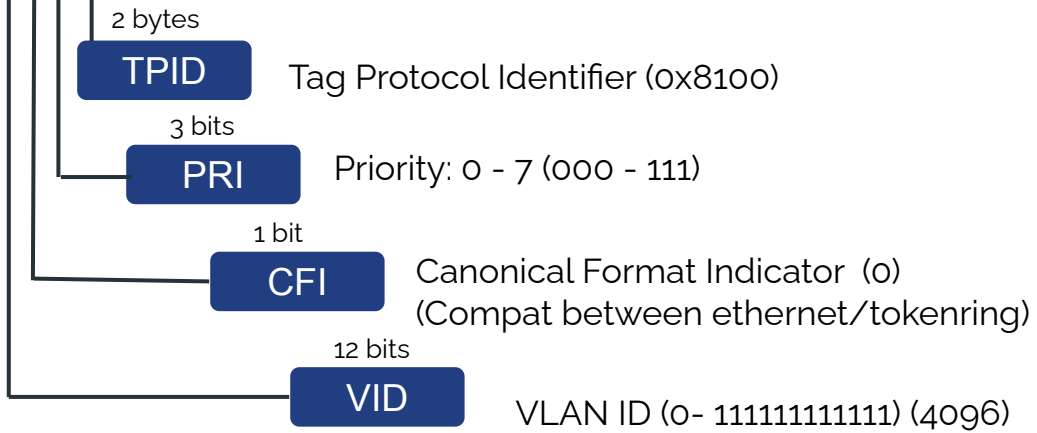
Advance concept (VLAN)

Conventional Ethernet frame format

6 bytes	6 bytes	2 bytes	46 - 1500 bytes	4 bytes
Destination address	Source address	Length/Type	Data	FCS

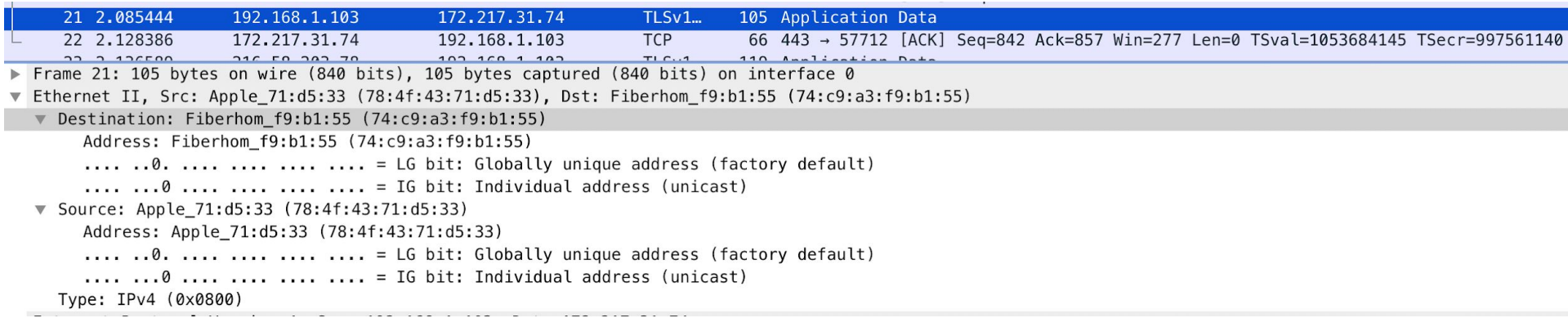
802.1Q frame format

6 bytes	6 bytes	4 bytes	2 bytes	46 - 1500 bytes	4 bytes
Destination address	Source address	802.1Q Tag	Length/Type	Data	FCS



Advance concept (VLAN)

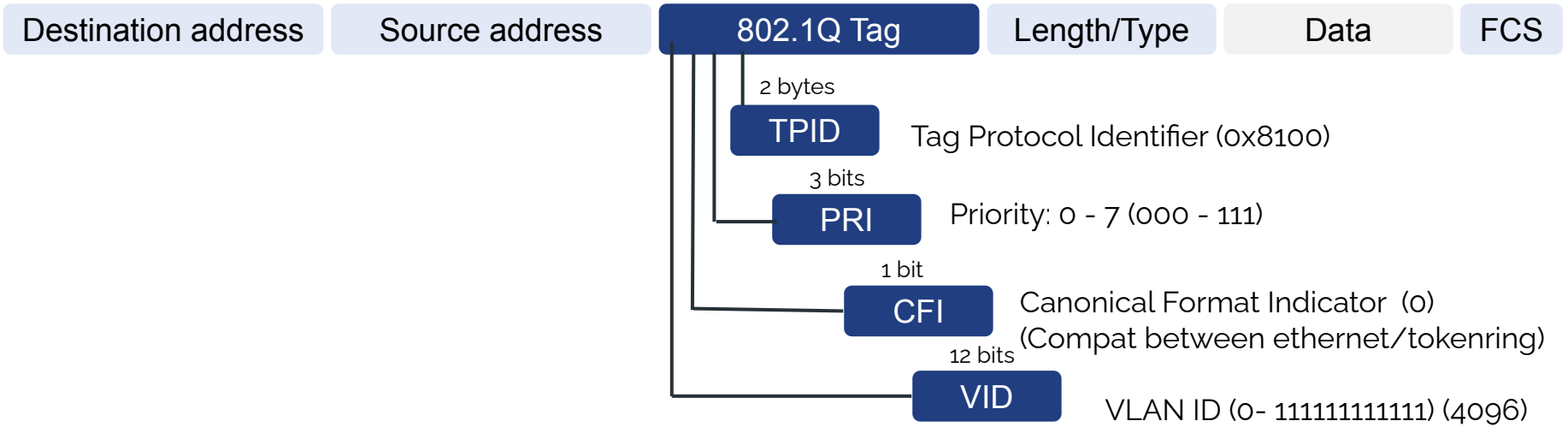
Conventional Ethernet frame format



Advance concept (VLAN)

802.1Q frame format

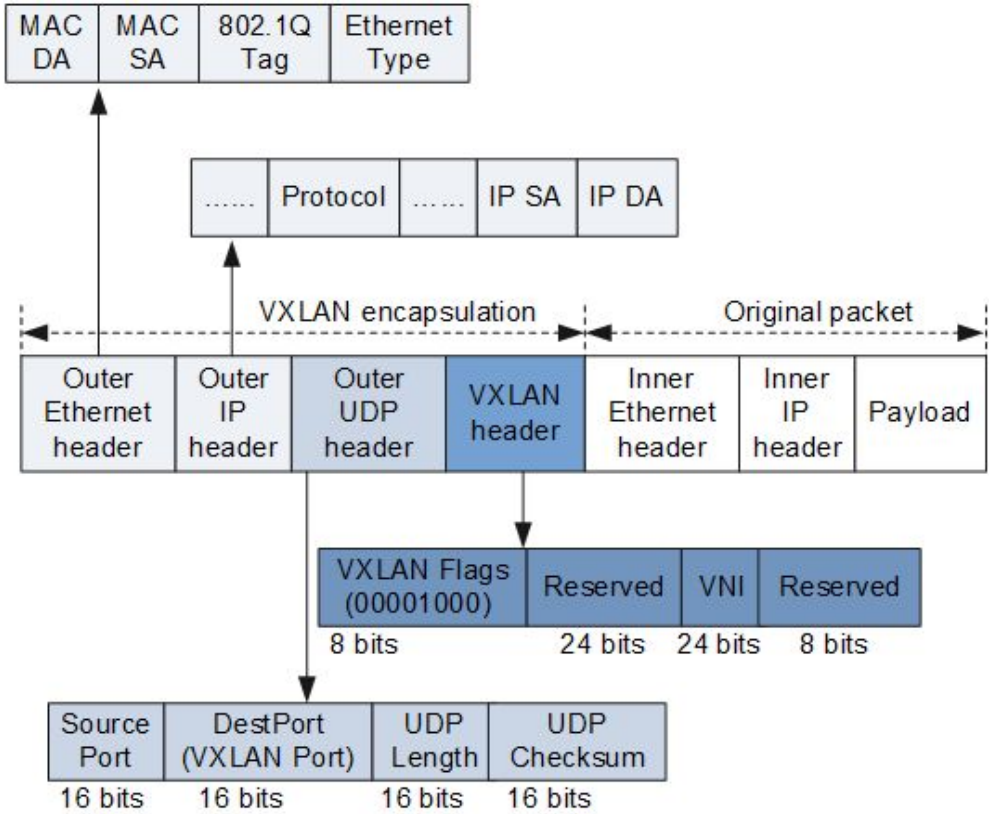
```
▷ Frame 6 (50 bytes on wire, 50 bytes captured)
▽ Ethernet II, Src: 3com_03:04:05 (00:01:02:03:04:05), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ▷ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ▷ Source: 3com_03:04:05 (00:01:02:03:04:05)
    Type: 802.1Q Virtual LAN (0x8100)
▽ 802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 1
    000. .... .... .... = Priority: 0
    ...0 .... .... .... = CFI: 0
    .... 0000 0000 0001 = ID: 1
    Type: 802.1Q Virtual LAN (0x8100)
▽ 802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 10
    000. .... .... .... = Priority: 0
    ...0 .... .... .... = CFI: 0
    .... 0000 0000 1010 = ID: 10
    Type: IP (0x0800)
▷ Internet Protocol, Src: 192.168.0.1 (192.168.0.1), Dst: 255.255.255.255 (255.255.255.255)
▷ Internet Control Message Protocol
```



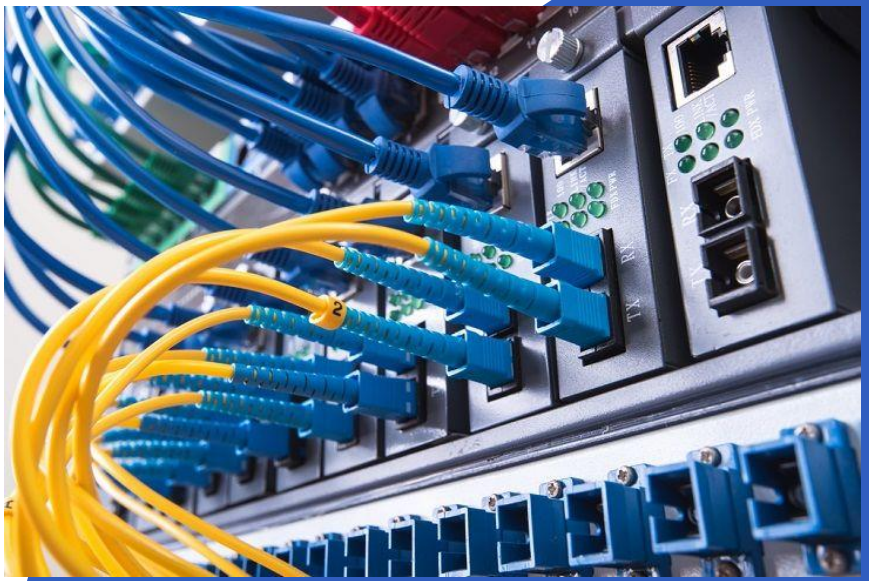
Advance concept (VXLAN)

- VXLAN (Virtual LAN) (RFC:7348)
- VXLAN is full network virtualization for extend capability of VLAN (L2) across L4 network (UDP)
- From architecture design above. VXLAN can distribute VLAN across L4 network layer with scalability up to 16 million network
- VXLAN concept original create by VMWare, Arista Network and Cisco
- Now VXLAN is standard software define network that use on Docker Swarm, Kubernetes and many open source project.

Advance concept (VXLAN)



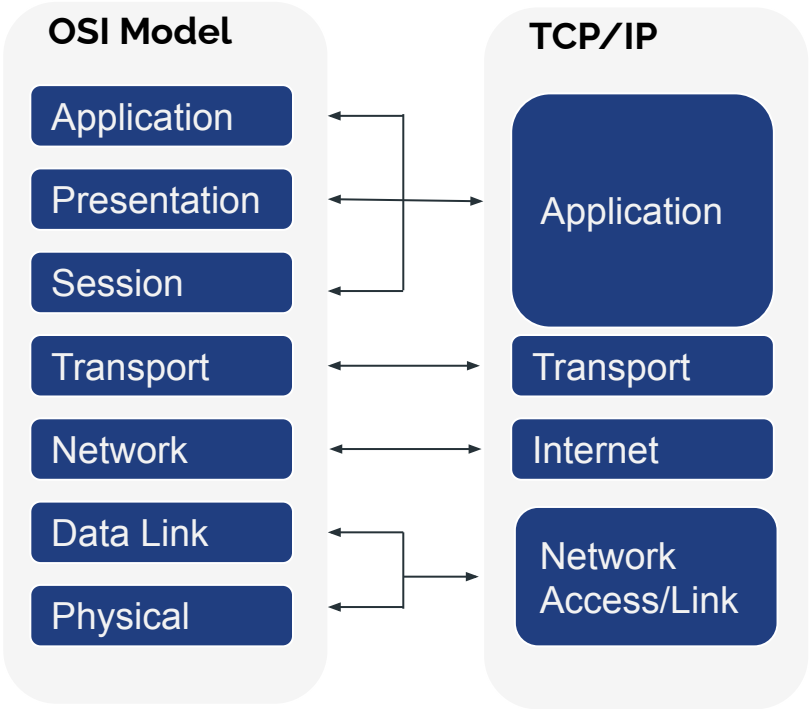
TCP/IP Architecture

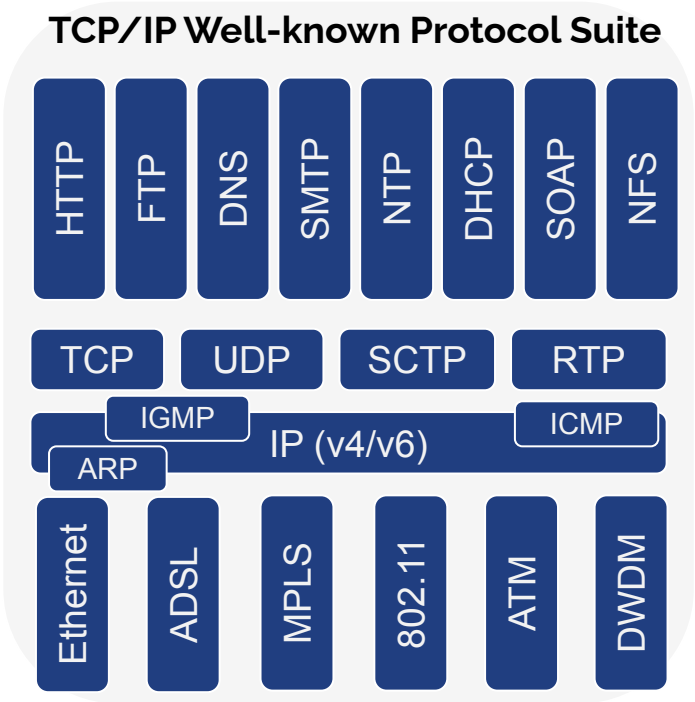


TCP/IP Architecture

- Develop by Defense Advanced Research Projects Agency (DARPA) since 1969
- Well-known as Department of Defense (US), IBM, AT&T adapt to computer network standard on 1982 - 1985
- Similar with OSI Model but conclude necessary layer together with 4 layer
    - Network Access/Link (L1)
    - Internet (L2)
    - Transport (L3)
    - Application (L4)
- The funny is everyone use TCP/IP for network communication. But as TCP/IP is very rough in term of role and responsibility for each layer. On practice we also reference OSI model for design and operate network communication

Reflect architecture between OSI and TCP/IP

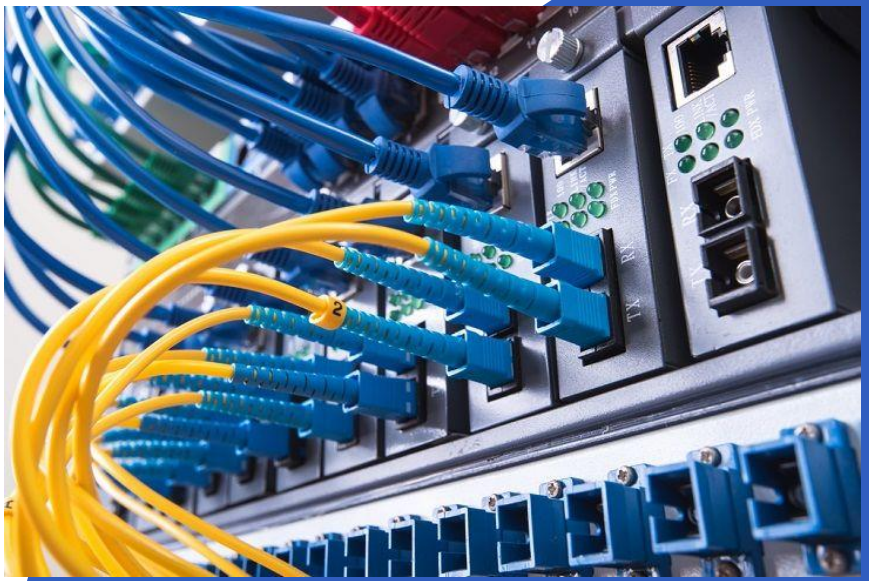




Workshop 1.1 Access Cloud

```
praparns-MacBook-Pro:.ssh praparn$ ssh -i "lab_gluster.pem" ubuntu@52.221.214.45
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1029-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat Dec 26 15:56:32 UTC 2020

  System load:  0.0               Processes:           93
  Usage of /:   14.8% of 7.69GB   Users logged in:     1
  Memory usage: 18%               IP address for eth0: 10.21.2.237
  Swap usage:   0%


0 packages can be updated.
0 updates are security updates.

New release '20.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Sat Dec 26 14:55:27 2020 from 171.100.88.179
ubuntu@ip-10-21-2-237:~$ ip -br -c addr show
lo               UNKNOWN        127.0.0.1/8 ::1/128
eth0             UP             10.21.2.237/24 fe80::411:ccff:fe15:3104/64
ubuntu@ip-10-21-2-237:~$
ubuntu@ip-10-21-2-237:~$ ip -br -c link show
lo               UNKNOWN        00:00:00:00:00:00 <LOOPBACK,UP,LOWER_UP>
eth0             UP             06:11:cc:15:31:04 <BROADCAST,MULTICAST,UP,LOWER_UP>
ubuntu@ip-10-21-2-237:~$
ubuntu@ip-10-21-2-237:~$ ip r
default via 10.21.2.1 dev eth0 proto dhcp src 10.21.2.237 metric 100
10.21.2.0/24 dev eth0 proto kernel scope link src 10.21.2.237
10.21.2.1 dev eth0 proto dhcp scope link src 10.21.2.237 metric 100
ubuntu@ip-10-21-2-237:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.21.2.1       0.0.0.0         UG        0 0          0 eth0
10.21.2.0       0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.21.2.1       0.0.0.0         255.255.255.255 UH        0 0          0 eth0
ubuntu@ip-10-21-2-237:~$
```

MAC Address Operation



ARP operation

- MAC Address is on OSI L2 layer that design for communication between host on same network
- This scenario it will start communicate via L2 layer (Ethernet: MAC address), IP Address will communicate via mac address
- This process will start operate with ARP (Address resolution protocol) process
  - Step1: Check destination ip with arp cache (Available ?)
  - Step2:
    - If existing to arp cache, Send via mac address on cache (end)
    - If not exist, Send **ARP req** (Who have mac address for ip: xxxx, Please reply)
  - Step3: **ARP req** will broadcast to all host on L2 (switch, ap, host)
  - Step4: Dest host who have ip address will reply **ARP reply** back
  - Step5: Source got ARP reply and send packet via mac address

ARP operation

Octet offset	0	1
0	Hardware type (HTYPE)	
2	Protocol type (PTYPE)	
4	Hardware address length (HLEN)	Protocol address length (PLEN)
6	Operation (OPER)	
8	Sender hardware address (SHA) (first 2 bytes)	
10	(next 2 bytes)	
12	(last 2 bytes)	
14	Sender protocol address (SPA) (first 2 bytes)	
16	(last 2 bytes)	
18	Target hardware address (THA) (first 2 bytes)	
20	(next 2 bytes)	
22	(last 2 bytes)	
24	Target protocol address (TPA) (first 2 bytes)	
26	(last 2 bytes)	

ARP operation

- **Example**:
  - Source: 10.101.200.250 (mac address: 00-0d-56-b9-21-20)
  - Destination: 10.101.200.100 (mac address: 00-01-23-a9-00-f3)
- Step1: Check arp cache for 10.101.200.100 (Result: no cache)
- Step2: Send arp request
  - Opcode:(0x0001)(Request)
  - Sender ip: 10.101.200.250
  - Sender mac: 00-0d-56-b9-21-20
  - Target ip: 10.101.200.100
  - Target mac: ff-ff-ff-ff-ff-ff
- Step3: arp request was sent to all host on L2 network (All host get arp cache for ip: 10.101.200.100 and mac: 00-0d-56-b9-21-20)

ARP operation

- **Example**:(Continue)
- Step4: Host who have ip address: 10.101.200.100 will send arp reply
  - Opcode: (0x0010)(reply)
  - Sender ip: 10.101.200.100
  - Sender mac: 00-01-23-a9-00-f3
  - Target ip: 10.101.200.250
  - Target mac: 00-0d-56-b9-21-20
- Step5: Sender will start communicate with destination from information of reply packet
- *Remark:
  - All device on L2 will learn on arp cache for packet arp request
  - Switch L2, will record arp cache when have host send/receive packet via port

ARP operation

A: Arp Req: Who has x.x.x.x ?please tell: x.x.x.x
Sender IP: 10.101.200.250
Sender Mac: 00-0d-56-b9-21-20
Dest IP: 10.101.200.100
Dest Mac: 00-00-00-00-00

Default:
10.101.200.60

B: Arp Reply: x.x.x.x is at : x.x.x.x
Sender IP: 10.101.200.100
Sender Mac: 00-01-23-a9-00-f4
Dest IP: 10.101.200.250
Dest Mac: 00-0d-56-b9-21-20

Port 10

Switch L2 Connection

Port 01

A: Arp Rep

Computer A:
10.101.200.250
00-0d-56-b9-21-20

Port 02

A: Arp Rep

Computer B:
10.101.200.12
00-0d-56-b9-23-fa

Port 03

A: Arp Rep

A: Arp Reply

Computer C:
10.101.200.100
00-01-23-a9-00-f3

Port 04

A: Arp Rep

Computer D:
10.101.200.230
00-01-23-a9-98-20

Workshop 1.2 ARP/IP Operation

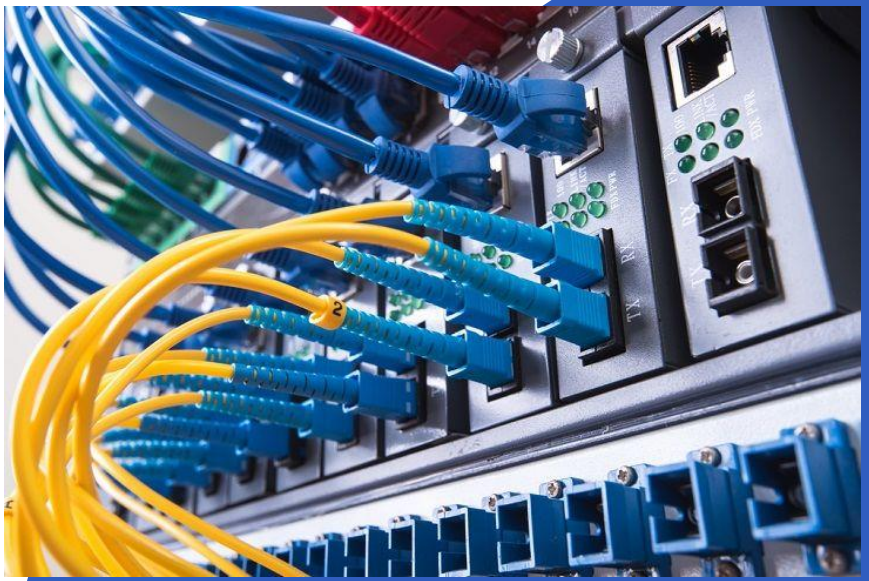
```
ubuntu@ip-10-21-2-237:~$ sudo ip -s -s neigh flush all
10.21.2.22 dev eth0 lladdr 06:36:ae:aa:e3:78 used 354/354/323 probes 4 STALE
10.21.2.1 dev eth0 lladdr 06:4d:1e:bf:fe:8e ref 1 used 50/0/49 probes 1 REACHABLE

*** Round 1, deleting 2 entries ***
10.21.2.1 dev eth0  ref 1 used 0/60/0 probes 4 INCOMPLETE

*** Round 2, deleting 1 entries ***
10.21.2.1 dev eth0 lladdr 06:4d:1e:bf:fe:8e ref 1 used 0/0/0 probes 4 REACHABLE

*** Round 3, deleting 1 entries ***
*** Flush is complete after 3 rounds ***
ubuntu@ip-10-21-2-237:~$ sudo tcpdump -enni eth0 arp -c 4
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
15:22:10.416107 06:11:cc:15:31:04 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Request who-has 10.21.2.22 tell 10.21.2.237, length 28
15:22:10.416232 06:36:ae:aa:e3:78 > 06:11:cc:15:31:04, ethertype ARP (0x0806), length 56: Reply 10.21.2.22 is-at 06:36:ae:aa:e3:78, length 42
15:22:15.696036 06:4d:1e:bf:fe:8e > 06:11:cc:15:31:04, ethertype ARP (0x0806), length 56: Request who-has 10.21.2.237 tell 10.21.2.1, length 42
15:22:15.696063 06:11:cc:15:31:04 > 06:4d:1e:bf:fe:8e, ethertype ARP (0x0806), length 42: Reply 10.21.2.237 is-at 06:11:cc:15:31:04, length 28
4 packets captured
4 packets received by filter
0 packets dropped by kernel
ubuntu@ip-10-21-2-237:~$ ip neigh
10.21.2.22 dev eth0 lladdr 06:36:ae:aa:e3:78 REACHABLE
10.21.2.1 dev eth0 lladdr 06:4d:1e:bf:fe:8e REACHABLE
ubuntu@ip-10-21-2-237:~$
```

```
ubuntu@ip-10-21-2-237:~$ ssh ubuntu@10.21.2.22 -v
OpenSSH_7.6p1 Ubuntu-4ubuntu0.3, OpenSSL 1.0.2n  7 Dec 2017
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: /etc/ssh/ssh_config line 19: Applying options for *
debug1: Connecting to 10.21.2.22 [10.21.2.22] port 22.
debug1: Connection established.
debug1: key_load_public: No such file or directory
debug1: identity file /home/ubuntu/.ssh/id_rsa type -1
debug1: key_load_public: No such file or directory
debug1: identity file /home/ubuntu/.ssh/id_rsa-cert type -1
debug1: key_load_public: No such file or directory
debug1: identity file /home/ubuntu/.ssh/id_dsa type -1
debug1: key_load_public: No such file or directory
debug1: identity file /home/ubuntu/.ssh/id_dsa-cert type -1
debug1: key_load_public: No such file or directory
debug1: identity file /home/ubuntu/.ssh/id_ecdsa type -1
debug1: key_load_public: No such file or directory
debug1: identity file /home/ubuntu/.ssh/id_ecdsa-cert type -1
debug1: key_load_public: No such file or directory
debug1: identity file /home/ubuntu/.ssh/id_ed25519 type -1
debug1: key_load_public: No such file or directory
debug1: identity file /home/ubuntu/.ssh/id_ed25519-cert type -1
debug1: Local version string SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3
debug1: Remote protocol version 2.0, remote software version OpenSSH_7.6p1 Ubuntu-4ubuntu0.3
debug1: match: OpenSSH_7.6p1 Ubuntu-4ubuntu0.3 pat OpenSSH* compat 0x04000000
debug1: Authenticating to 10.21.2.22:22 as 'ubuntu'
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: algorithm: curve25519-sha256
debug1: kex: host key algorithm: ecdsa-sha2-nistp256
debug1: kex: server->client cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
debug1: kex: client->server cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
debug1: expecting SSH2_MSG_KEX_ECDH_REPLY
debug1: Server host key: ecdsa-sha2-nistp256 SHA256:J279ANkSWMupDco1xyrhQlXLyLtWoE6KoGrjL5EH68Q
debug1: Host '10.21.2.22' is known and matches the ECDSA host key.
debug1: Found key in /home/ubuntu/.ssh/known_hosts:1
debug1: rekey after 134217728 blocks
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: rekey after 134217728 blocks
debug1: SSH2_MSG_EXT_INFO received
debug1: kex_input_ext_info: server-sig-algs=<ssh-ed25519,ssh-rsa,rsa-sha2-256,rsa-sha2-512,ssh-dss,e
debug1: SSH2_MSG_SERVICE_ACCEPT received
debug1: Authentications that can continue: publickey
debug1: Next authentication method: publickey
debug1: Trying private key: /home/ubuntu/.ssh/id_rsa
debug1: Trying private key: /home/ubuntu/.ssh/id_dsa
debug1: Trying private key: /home/ubuntu/.ssh/id_ecdsa
debug1: Trying private key: /home/ubuntu/.ssh/id_ed25519
debug1: No more authentication methods to try.
ubuntu@10.21.2.22: Permission denied (publickey).
ubuntu@ip-10-21-2-237:~$
```

Workshop 1.2 ARP/IP Operation

```
ubuntu@ip-10-21-2-237:~$ sudo ip -s -s neigh flush all
10.21.2.22 dev eth0 lladdr 06:36:ae:aa:e3:78 ref 1 used 13/13/13 probes 4 REACHABLE
10.21.2.1 dev eth0 lladdr 06:4d:1e:bf:fe:8e ref 1 used 89/0/89 probes 4 REACHABLE

*** Round 1, deleting 2 entries ***
10.21.2.1 dev eth0  ref 1 used 0/60/0 probes 4 INCOMPLETE

*** Round 2, deleting 1 entries ***
10.21.2.1 dev eth0 lladdr 06:4d:1e:bf:fe:8e ref 1 used 0/0/0 probes 4 REACHABLE

*** Round 3, deleting 1 entries ***
*** Flush is complete after 3 rounds ***
ubuntu@ip-10-21-2-237:~$ sudo tcpdump -enni eth0 host 10.21.2.22 or arp -c 100
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
15:32:10.145092 06:4d:1e:bf:fe:8e > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 56: Request who-has 10.21.2.237 tell 10.21.2.1, length 42
15:32:10.145120 06:11:cc:15:31:04 > 06:4d:1e:bf:fe:8e, ethertype ARP (0x0806), length 42: Reply 10.21.2.237 is-at 06:11:cc:15:31:04, length 28
15:32:11.227592 06:11:cc:15:31:04 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Request who-has 10.21.2.22 tell 10.21.2.237, length 28
15:32:11.227747 06:36:ae:aa:e3:78 > 06:11:cc:15:31:04, ethertype ARP (0x0806), length 56: Reply 10.21.2.22 is-at 06:36:ae:aa:e3:78, length 42
15:32:11.227755 06:11:cc:15:31:04 > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 74: 10.21.2.237.44516 > 10.21.2.22.22: Flags [S], seq 1198964702, win 62727, options [mss 8961,sackOK,TS val 715325114 ecr 0,nop,wscale 7], length 0
15:32:11.228158 06:36:ae:aa:e3:78 > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 74: 10.21.2.22.22 > 10.21.2.237.44516: Flags [S.], seq 3088878284, ack 1198964703, win 62643, options [mss 8961,sackOK,TS val 2508344577 ecr 7153251
4,nop,wscale 7], length 0
15:32:11.228174 06:11:cc:15:31:04 > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 66: 10.21.2.237.44516 > 10.21.2.22.22: Flags [.], ack 1, win 491, options [nop,nop,TS val 715325115 ecr 2508344577], length 0
15:32:11.229190 06:11:cc:15:31:04 > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 107: 10.21.2.237.44516 > 10.21.2.22.22: Flags [P.], seq 1:42, ack 1, win 491, options [nop,nop,TS val 715325116 ecr 2508344577], length 41
15:32:11.229504 06:36:ae:aa:e3:78 > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 66: 10.21.2.22.22 > 10.21.2.237.44516: Flags [.], ack 42, win 490, options [nop,nop,TS val 2508344578 ecr 715325116], length 0
15:32:11.233428 06:36:ae:aa:e3:78 > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 107: 10.21.2.22.22 > 10.21.2.237.44516: Flags [P.], seq 1:42, ack 42, win 490, options [nop,nop,TS val 2508344582 ecr 715325116], length 41
15:32:11.233438 06:11:cc:15:31:04 > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 66: 10.21.2.237.44516 > 10.21.2.22.22: Flags [.], ack 42, win 491, options [nop,nop,TS val 715325120 ecr 2508344582], length 0
15:32:11.234023 06:36:ae:aa:e3:78 > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 1146: 10.21.2.22.22 > 10.21.2.237.44516: Flags [P.], seq 42:1122, ack 42, win 490, options [nop,nop,TS val 2508344583 ecr 715325120], length 1080
15:32:11.234032 06:11:cc:15:31:04 > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 66: 10.21.2.237.44516 > 10.21.2.22.22: Flags [.], ack 1122, win 483, options [nop,nop,TS val 715325121 ecr 2508344583], length 0
15:32:11.234148 06:11:cc:15:31:04 > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 1426: 10.21.2.237.44516 > 10.21.2.22.22: Flags [P.], seq 42:1402, ack 1122, win 483, options [nop,nop,TS val 715325121 ecr 2508344583], length 1360
15:32:11.234480 06:36:ae:aa:e3:78 > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 66: 10.21.2.22.22 > 10.21.2.237.44516: Flags [.], ack 1402, win 480, options [nop,nop,TS val 2508344583 ecr 715325121], length 0
15:32:11.236677 06:11:cc:15:31:04 > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 114: 10.21.2.237.44516 > 10.21.2.22.22: Flags [P.], seq 1402:1450, ack 1122, win 483, options [nop,nop,TS val 715325123 ecr 2508344583], length 48
15:32:11.236954 06:36:ae:aa:e3:78 > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 66: 10.21.2.22.22 > 10.21.2.237.44516: Flags [.], ack 1450, win 480, options [nop,nop,TS val 2508344586 ecr 715325123], length 0
15:32:11.241301 06:36:ae:aa:e3:78 > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 518: 10.21.2.22.22 > 10.21.2.237.44516: Flags [P.], seq 1122:1574, ack 1450, win 480, options [nop,nop,TS val 2508344590 ecr 715325123], length 452
15:32:11.241311 06:11:cc:15:31:04 > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 66: 10.21.2.237.44516 > 10.21.2.22.22: Flags [.], ack 1574, win 480, options [nop,nop,TS val 715325128 ecr 2508344590], length 0
15:32:11.244348 06:11:cc:15:31:04 > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 82: 10.21.2.237.44516 > 10.21.2.22.22: Flags [P.], seq 1450:1466, ack 1574, win 480, options [nop,nop,TS val 715325131 ecr 2508344590], length 16
15:32:11.244665 06:36:ae:aa:e3:78 > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 66: 10.21.2.22.22 > 10.21.2.237.44516: Flags [.], ack 1466, win 480, options [nop,nop,TS val 2508344593 ecr 715325131], length 0
15:32:11.244675 06:11:cc:15:31:04 > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 110: 10.21.2.237.44516 > 10.21.2.22.22: Flags [P.], seq 1466:1510, ack 1574, win 480, options [nop,nop,TS val 715325131 ecr 2508344593], length 44
15:32:11.244950 06:36:ae:aa:e3:78 > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 66: 10.21.2.22.22 > 10.21.2.237.44516: Flags [.], ack 1510, win 480, options [nop,nop,TS val 2508344594 ecr 715325131], length 0
15:32:11.244997 06:36:ae:aa:e3:78 > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 110: 10.21.2.22.22 > 10.21.2.237.44516: Flags [P.], seq 1574:1618, ack 1510, win 480, options [nop,nop,TS val 2508344594 ecr 715325131], length 44
15:32:11.245001 06:11:cc:15:31:04 > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 66: 10.21.2.237.44516 > 10.21.2.22.22: Flags [.], ack 1618, win 480, options [nop,nop,TS val 715325132 ecr 2508344594], length 0
15:32:11.245158 06:11:cc:15:31:04 > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 134: 10.21.2.237.44516 > 10.21.2.22.22: Flags [P.], seq 1510:1578, ack 1618, win 480, options [nop,nop,TS val 715325132 ecr 2508344594], length 68
15:32:11.245440 06:36:ae:aa:e3:78 > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 66: 10.21.2.22.22 > 10.21.2.237.44516: Flags [.], ack 1578, win 480, options [nop,nop,TS val 2508344594 ecr 715325132], length 0
15:32:11.247070 06:36:ae:aa:e3:78 > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 110: 10.21.2.22.22 > 10.21.2.237.44516: Flags [P.], seq 1618:1662, ack 1578, win 480, options [nop,nop,TS val 2508344596 ecr 715325132], length 44
15:32:11.247079 06:11:cc:15:31:04 > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 66: 10.21.2.237.44516 > 10.21.2.22.22: Flags [.], ack 1662, win 480, options [nop,nop,TS val 715325134 ecr 2508344596], length 0
15:32:11.247689 06:11:cc:15:31:04 > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 66: 10.21.2.237.44516 > 10.21.2.22.22: Flags [F.], seq 1578, ack 1662, win 480, options [nop,nop,TS val 715325134 ecr 2508344596], length 0
15:32:11.249258 06:36:ae:aa:e3:78 > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 66: 10.21.2.22.22 > 10.21.2.237.44516: Flags [F.], seq 1662, ack 1579, win 480, options [nop,nop,TS val 2508344598 ecr 715325134], length 0
15:32:11.249269 06:11:cc:15:31:04 > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 66: 10.21.2.237.44516 > 10.21.2.22.22: Flags [.], ack 1663, win 480, options [nop,nop,TS val 715325136 ecr 2508344598], length 0
15:32:15.275802 06:4d:1e:bf:fe:8e > 06:11:cc:15:31:04, ethertype ARP (0x0806), length 56: Request who-has 10.21.2.237 tell 10.21.2.1, length 42
15:32:15.275828 06:11:cc:15:31:04 > 06:4d:1e:bf:fe:8e, ethertype ARP (0x0806), length 42: Reply 10.21.2.237 is-at 06:11:cc:15:31:04, length 28
^C
34 packets captured
34 packets received by filter
0 packets dropped by kernel
ubuntu@ip-10-21-2-237:~$
```

IP Address and Routing



Mathematic and IP Address Overview

- Computer network is pure calculation for operate by binary base
- As normally decimal base are familiar than binary base.
- To make better understand how computer calculate and operate on network. We will introduce some basic concept of mathematic for network communication.
    - Conversion between decimal, binary, hexadecimal
    - Basic/Advance operation

Decimal Base	Binay Base	Hexadecimal Base
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Conversion between base number

- All base number have the position start from least significant digit (**lsd**) to most significant digit (**msd**)

	MSD			**Decimal (Ex: 243 ⇒200 + 40 + 3)**						**LSD**
	10^{10}	10^{9}	10^{8}	10^{7}	10^{6}	10^{5}	10^{4}	10^{3}	10^{2}	10^{1}	10^{0}
Value	-	-	-	-	-	-	-	-	2	4	3

	MSD			**Binary (Ex: 243 ⇒ 128 + 64 + 32 + 16 +2 +1)**						**LSD**
	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Position	1024	512	256	128	64	32	16	8	4	2	1
Value	-	-	-	1	1	1	1	0	0	1	1

Conversion between base number

MSD **Binary (Ex: 243 ⇒ 128 + 64 + 32 + 16 +2 +1)** **LSD**

	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Position	1024	512	256	128	64	32	16	8	4	2	1
Value	-	-	-	1	1	1	1	0	0	1	1

Group of 4 digits Group of 4 digits

MSD **Hexadecimal (Ex: 1111 ⇒ F, 0011 ⇒ 3)** **LSD**

	16^{10}	16^9	16^8	16^7	16^6	16^5	16^4	16^3	16^2	16^1	16^0
Position	-	-	-	-	-	-	65536	4096	256	16	1
Value	-	-	-	-	-	-	-	-	-	F	3

Conversion between base number

MSD — Hexadecimal (Ex: 1111 ⇒ F, 0011 ⇒ 3) — LSD

	16^{10}	16^{9}	16^{8}	16^{7}	16^{6}	16^{5}	16^{4}	16^{3}	16^{2}	16^{1}	16^{0}
Position	-	-	-	-	-	-	65536	4096	256	16	1
Value	-	-	-	-	-	-	-	-	-	F	3

Group of 4 digits

Group of 4 digits

	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Position	1024	512	256	128	64	32	16	8	4	2	1
Value	-	-	-	1	1	1	1	0	0	1	1

Practice 1.1: Basic conversion

- Please convert 125 (decimal) to binary and hexadecimal
- Ex1: Decimal ⇒ Binary

	MSD				Decimal (Ex: 125 ⇒100 + 20 + 5)						LSD
	10^{10}	10^{9}	10^{8}	10^{7}	10^{6}	10^{5}	10^{4}	10^{3}	10^{2}	10^{1}	10^{0}
Value	-	-	-	-	-	-	-	-	1	2	5

	MSD				Binary (Ex: 125 ⇒ ?????????)						LSD
	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Position	1024	512	256	128	64	32	16	8	4	2	1
Value	-	-	-	-	-	-	-	-	-	-	-

Practice 1.1: Basic conversion

- Please convert 125 (decimal) to binary and hexadecimal
- Ex1: Decimal ⇒ Binary

MSD **Binary (Ex: 125 ⇒ 64 + 32 ⇒ 96)** **LSD**

	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Position	1024	512	256	128	64	32	16	8	4	2	1
Value	-	-	-	-	1	1	-	-	-	-	-

MSD **Binary (Ex: 125 ⇒ 96 + 16 ⇒ 112)** **LSD**

	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Position	1024	512	256	128	64	32	16	8	4	2	1
Value	-	-	-	-	1	1	1	-	-	-	-

Practice 1.1: Basic conversion

- Please convert 125 (decimal) to binary and hexadecimal
- Ex1: Decimal ⇒ Binary

MSD **Binary (Ex: 125 ⇒ 112 + 8 ⇒ 120)** **LSD**

	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Position	1024	512	256	128	64	32	16	8	4	2	1
Value	-	-	-	-	1	1	1	1	-	-	-

MSD **Binary (Ex: 125 ⇒ 120 + 5 (4+1)) ⇒ 1111101** **LSD**

	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Position	1024	512	256	128	64	32	16	8	4	2	1
Value	-	-	-	-	1	1	1	1	1	0	1

Practice 1.1: Basic conversion

- OR ?

MSD **Binary (Ex: 127 (111111) - 125 = 2)** **LSD**

	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Position	1024	512	256	128	64	32	16	8	4	2	1
Value	-	-	-	-	1	1	1	1	1	1	1

MSD **Binary (Ex: 125 ⇒ 1111101)** **LSD**

	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Position	1024	512	256	128	64	32	16	8	4	2	1
Value	-	-	-	-	1	1	1	1	1	0	1

Practice 1.1: Basic conversion

- Please convert 125 (decimal) to binary and hexadecimal
- Ex1: Binary ⇒ Hexadecimal

MSD **Binary (Ex: 125 ⇒ 1111101)** **LSD**

	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Position	1024	512	256	128	64	32	16	8	4	2	1
Value	-	-	-	-	1	1	1	1	1	0	1

Group of 4 digits ⇒ 1101 ⇒ D (13)

Group of 4 digits ⇒ 111 ⇒ 7

MSD **Hexadecimal (Ex: 0111 ⇒ 7, 1101 ⇒D (13))** **LSD**

	16^{10}	16^9	16^8	16^7	16^6	16^5	16^4	16^3	16^2	16^1	16^0
Position	-	-	-	-	-	-	65536	4096	256	16	1
Value	-	-	-	-	-	-	-	-	-	7	D

Practice 1.1: Basic conversion

- Please convert 125 (decimal) to binary and hexadecimal
- Ex1: Decimal ⇒ Binary

MSD **Binary (Ex: 125 ⇒ 64 + 32 + 16 + 8 + 4 + 1)** **LSD**

	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Position	1024	512	256	128	64	32	16	8	4	2	1
Value	-	-	-	-	-	-	-	-	-	-	-

Practice 1.1: Basic conversion

- Please convert 203 (decimal) to binary and hexadecimal
- Ex2: Decimal ⇒ Binary

	MSD				Decimal (Ex: 203 ⇒200 + 3)						LSD
	10^{10}	10^{9}	10^{8}	10^{7}	10^{6}	10^{5}	10^{4}	10^{3}	10^{2}	10^{1}	10^{0}
Value	-	-	-	-	-	-	-	-	2	0	3

	MSD				Binary (Ex: 203 ⇒ ?????????)						LSD
	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Position	1024	512	256	128	64	32	16	8	4	2	1
Value	-	-	-	-	-	-	-	-	-	-	-

Practice 1.1: Basic conversion

- Please convert 203 (decimal) to binary and hexadecimal
- Ex2: Decimal ⇒ Binary

MSD **Binary (Ex: 203 ⇒ 128 + 64 ⇒ 192)** **LSD**

	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Position	1024	512	256	128	64	32	16	8	4	2	1
Value	-	-	-	1	1	-	-	-	-	-	-

MSD **Binary (Ex: 203 ⇒ 192 + 8 ⇒ 200)** **LSD**

	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Position	1024	512	256	128	64	32	16	8	4	2	1
Value	-	-	-	1	1	0	0	1	-	-	-

Practice 1.1: Basic conversion

- Please convert 203 (decimal) to binary and hexadecimal
- Ex2: Decimal ⇒ Binary

	MSD				**Binary (Ex: 203 ⇒ 200 + 2 + 1 ⇒ 203)**						**LSD**
	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Position	1024	512	256	128	64	32	16	8	4	2	1
Value	-	-	-	1	1	0	0	1	0	1	1

Practice 1.1: Basic conversion

- Please convert 203 (decimal) to binary and hexadecimal
- Ex2: Binary ⇒ Hexadecimal

MSD **Binary (Ex: 203 ⇒ 128 + 64 + 8 + 2 + 1 ⇒ 203)** **LSD**

	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Position	1024	512	256	128	64	32	16	8	4	2	1
Value	-	-	-	1	1	0	0	1	0	1	1

Group of 4 digits ⇒ 1011 ⇒ B (11)

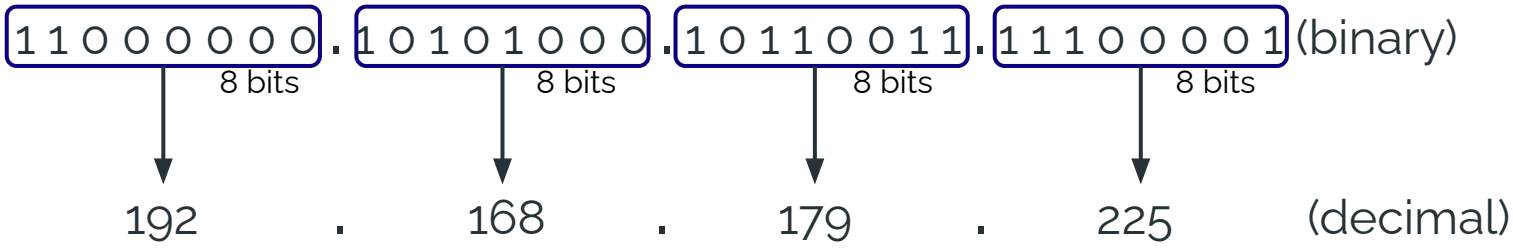
Group of 4 digits ⇒ 1100 ⇒ (C) 12

MSD **Hexadecimal (Ex: 0111 ⇒ 7, 1101 ⇒D (13))** **LSD**

	16^{10}	16^9	16^8	16^7	16^6	16^5	16^4	16^3	16^2	16^1	16^0
Position	-	-	-	-	-	-	65536	4096	256	16	1
Value	-	-	-	-	-	-	-	-	-	C	B

IP Address Overview

- IP Address also use concept of binary for operate with 32 bits sequential
- For easier to operate. IP Address will separate entire 32 bits to 4 segment (8 bits x 4)
- We usually written ip address on decimal base instead of binary base.
- Anyway please take note that on exactly calculation. Computer will operate by binary base operate only
- **Ex:**



IP Address Overview

- Convert 10.125.30.200 (decimal) to binary

10 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	**0**	**0**	**0**	**1**	**0**	**1**	**0**

125 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	**1**	**1**	**1**	**1**	**1**	**0**	**1**

30 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	**0**	**0**	**1**	**1**	**1**	**1**	**0**

200 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
1	**1**	**0**	**0**	**1**	**0**	**0**	**0**

00001010 . 01111101 . 00011110 . 11001000

IP Address Overview

- Convert 192.168.38.250 (decimal) to binary

192 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
1	**1**	**0**	**0**	**0**	**0**	**0**	**0**

168 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
1	**0**	**1**	**0**	**1**	**0**	**0**	**0**

38 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	**0**	**1**	**0**	**0**	**1**	**1**	**0**

250 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
1	**1**	**1**	**1**	**1**	**0**	**1**	**0**

11000000 . 10101000 . 00100110 . 11111010

Practice 1.2: IP conversion

- Ex1: Convert 172.16.200.198 (decimal) to binary

172 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
?	?	?	?	?	?	?	?

16 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
?	?	?	?	?	?	?	?

200 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
?	?	?	?	?	?	?	?

198 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
?	?	?	?	?	?	?	?

???????? . ???????? . ???????? . ????????

Practice 1.2: IP conversion

- Convert 172.16.200.198 (decimal) to binary

172 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
1	**0**	**1**	**0**	**1**	**1**	**0**	**0**

16 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	**0**	**0**	**1**	**0**	**0**	**0**	**0**

200 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
1	**1**	**0**	**0**	**1**	**0**	**0**	**0**

198 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
1	**1**	**0**	**0**	**0**	**1**	**1**	**0**

10101100 . 00010000 . 11001000 . 11000110

Practice 1.2: IP conversion

- Ex2: Convert 255.255.255.0 (decimal) to binary

255 (Decimal)

↓

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
?	**?**	**?**	**?**	**?**	**?**	**?**	**?**

255 (Decimal)

↓

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
?	**?**	**?**	**?**	**?**	**?**	**?**	**?**

255 (Decimal)

↓

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
?	**?**	**?**	**?**	**?**	**?**	**?**	**?**

0 (Decimal)

↓

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
?	**?**	**?**	**?**	**?**	**?**	**?**	**?**

???????? . ???????? . ???????? . ????????

Practice 1.2: IP conversion

- Convert 255.255.255.0 (decimal) to binary

255 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**

255 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**

255 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**

0 (Decimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**

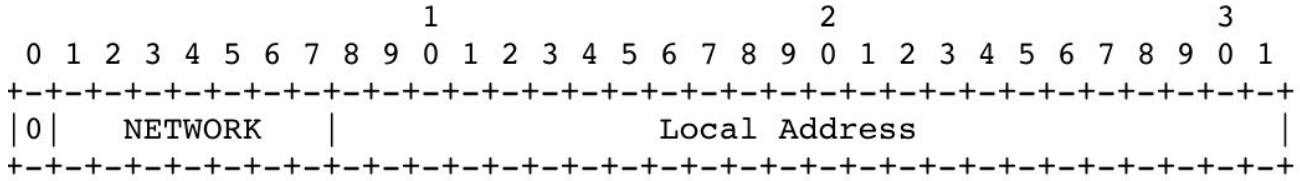
11111111 . 11111111 . 11111111 . 00000000

IP Address Structure and Class

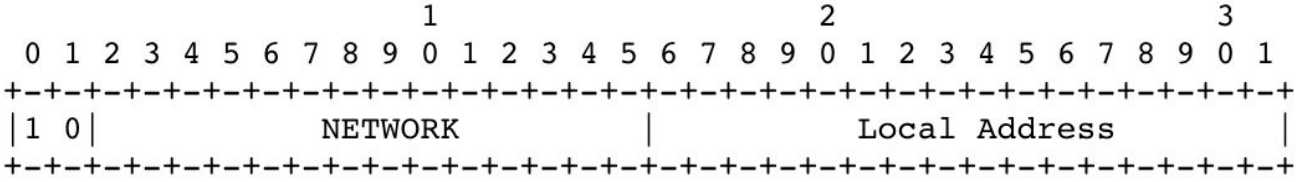
- IP Address structure was separated to 2 segment
  - Network segment: For separate network/routing
  - Host segment: Reserve for define host address in network
- For define IP Address in real-life. IETF are define standard class of network by IP Address (RFC 870, RFC 1466) for distribution IP Address in 3 class
- **Class A (Extra large assign)**
  - Network segment: 1 bit(0)+7 bits (126 networks)
  - Host segment: 24 bits (16,777,216 hosts)
- **Class B (Large assign)**
  - Network segment: 2 bits(10)+14 bits (16,384 networks)
  - Host segment: 16 bits (65,536 hosts)
- **Class C (Regular assign)**
  - Network segment: 3 bits(110)+21 bits (2,097,152 networks)
  - Host segment: 8 bits (254 hosts)

IP Address Structure and Class

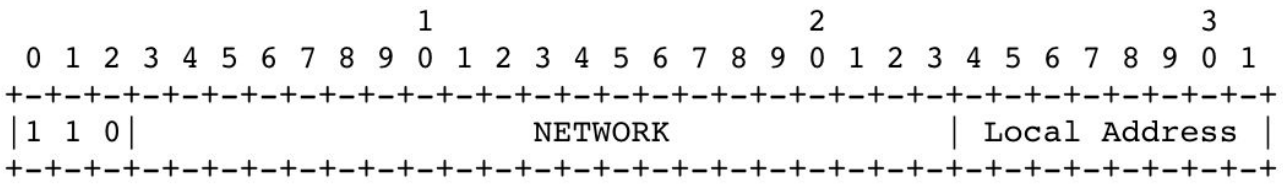
- Class A (0 - 127: 00000000 - 01111111)(first ip address segment)



- Class B (128 - 191: 10000000 - 10111111)(first ip address segment)



- Class C (192 - 223: 11000000 - 11011111)(first ip address segment)



IP Address Structure and Class

- **Special Class:**
- Class D (Multicast)
  - Network segment: 4 bits(1110)+24 bits
    - 224 - 239 (11100000 - 11101111)(first segment)
  - Host segment: 8 bits
  - Start: 224.0.0.0
  - Stop: 239.255.255.255
- Class E (IETF Reserve)
  - Network segment: 5 bits(11110)
    - 240 - 255 (11110000 - 11111111) (first segment)
  - Host segment: N/A
  - Start: 240.0.0.0
  - Stop: 255.255.255.255

IP Address Structure and Class

- **Reserve IP Address:**
  - 127.0.0.1: Loopback interface for host owner
  - 224.0.0.0: Multicast subnet
  - 255.255.255.255: All broadcast network
- IP Address allocation per regional (RFC 1466)

Multi-regional	192.0.0.0 - 193.255.255.255
Europe	194.0.0.0 - 195.255.255.255
Others	196.0.0.0 - 197.255.255.255
North America	198.0.0.0 - 199.255.255.255
Central/South America	200.0.0.0 - 201.255.255.255
Pacific Rim	202.0.0.0 - 203.255.255.255
Others	204.0.0.0 - 205.255.255.255
Others	206.0.0.0 - 207.255.255.255

Classless Inter-Domain Routing (CIDR)

- Another concept for define network/host on IP Address
- Static define class of IP Address to A, B, C that may not fit for some scenario
  - Class A is too large (16,777,216) but Class B is too small (65,536)
  - Huge of ip is unallocate for traditional design (Ex: 2,000 host with Class B)
  - This bring large broadcast on this design
- Instead to define static class. We will define network by how many bit that defined for network segment and other will be host segment
  - Ex1: 10.23.0.0/16 (network is 16 bits and host is 16 bits)
  - Ex2: 192.168.100.128/25 (network is 25 bits and host is 7 bits)
  - Ex3: 10.16.0.1/24 (network is 24 bits and host is 8 bits)
  - Ex4: 10.101.13.32/32 (network is 32 bit and host is 0 bits) (host route)

Classless Inter-Domain Routing (CIDR)

notation	addrs/block	# blocks	
n.n.n.n/32	1	4294967296	"host route"
n.n.n.x/31	2	2147483648	"p2p link"
n.n.n.x/30	4	1073741824	
n.n.n.x/29	8	536870912	
n.n.n.x/28	16	268435456	
n.n.n.x/27	32	134217728	
n.n.n.x/26	64	67108864	
n.n.n.x/25	128	33554432	
n.n.n.0/24	256	16777216	legacy "Class C"
n.n.x.0/23	512	8388608	
n.n.x.0/22	1024	4194304	
n.n.x.0/21	2048	2097152	
n.n.x.0/20	4096	1048576	
n.n.x.0/19	8192	524288	
n.n.x.0/18	16384	262144	
n.n.x.0/17	32768	131072	
n.n.0.0/16	65536	65536	legacy "Class B"
n.x.0.0/15	131072	32768	
n.x.0.0/14	262144	16384	
n.x.0.0/13	524288	8192	
n.x.0.0/12	1048576	4096	
n.x.0.0/11	2097152	2048	
n.x.0.0/10	4194304	1024	
n.x.0.0/9	8388608	512	
n.0.0.0/8	16777216	256	legacy "Class A"
x.0.0.0/7	33554432	128	
x.0.0.0/6	67108864	64	
x.0.0.0/5	134217728	32	
x.0.0.0/4	268435456	16	
x.0.0.0/3	536870912	8	
x.0.0.0/2	1073741824	4	
x.0.0.0/1	2147483648	2	
0.0.0.0/0	4294967296	1	"default route"

IP Private Addresses

- Normally public ip is global standard IP Address and it unique for identify host on internet network
- Any in real world public IP Address is never enough and always exhaustion !!!
- IETF was defined to resolve this problem for define some specific IP Address call “IP Private Address”. This IP Address range was defined as standard to non-routable on any internet backbone.
- Every organization can leverage this range of IP Address without any concern of duplicate on internet
- When any IP Private Address need to communication with internet. We will provide NAT (network address translation) for represent all private ip address to single public IP Address
- IP Private Address was defined class in RFC1918 to Class A, B, C

IP Private Addresses

- Class A:
  - Network segment: 8 bits (CIDR: /8)
    - Value: 10.x.x.x
    - Binary: **00001010**.x.x.x
    - Total network: 1 network
  - Host segment: 24 bits (16,77,216 hosts)
    - Value: 10.0.0.1 - 10.255.255.255
- Class B:
  - Network segment: 12 bits (CIDR: /12)
    - Value: 172.16.x.x - 172.31.x.x
    - Binary: **10101100.0001**0000.x.x - **10101100.0001**1111.x.x
    - Total network: 16 networks
  - Host segment: 20 bits (1,048,576 hosts)
    - Value: 172.16.0.1 - 172.31.255.255

IP Private Addresses

- Class C:
  - Network segment: 16 bits (CIDR: /16)
    - Value: 192.168.x.x
    - Binary: **11000000.10101000**.x.x - **11000000.10101000**.x.x
    - Total network: 1 network
  - Host segment: 16 bits
    - Value: 192.168.0.1 - 192.168.255.255
    - Total: 254 hosts

Private Address Space

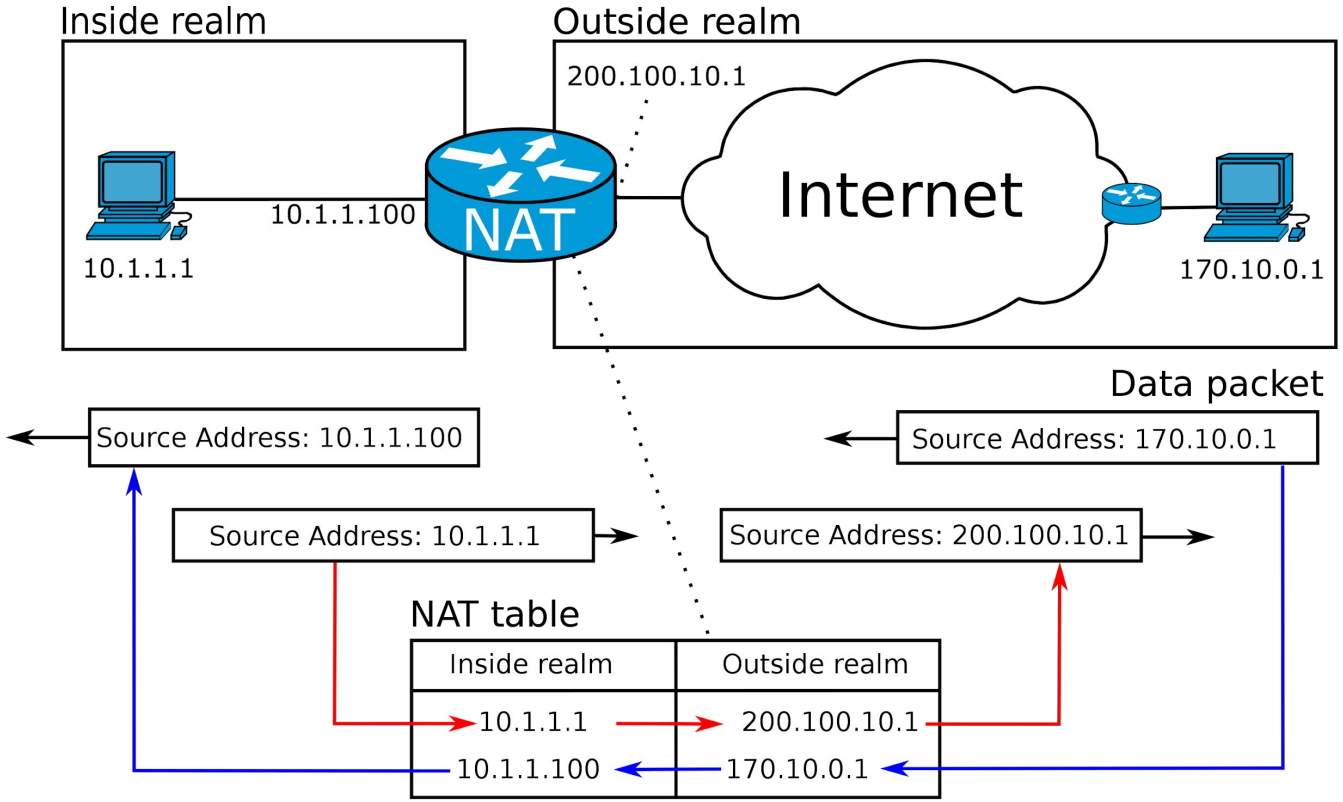
The Internet Assigned Numbers Authority (IANA) has reserved the following three blocks of the IP address space for private internets:

10.0.0.0	-	10.255.255.255	(10/8 prefix)
172.16.0.0	-	172.31.255.255	(172.16/12 prefix)
192.168.0.0	-	192.168.255.255	(192.168/16 prefix)

NAT Operation

- When all organization will use private ip address as assign. So how can their private ip address can communicate on internet ?
- NAT (Network Address Translation) is method to remapping original ip address to another ip address. This method can use to operate both inbound and outbound ip address
- Usually use for mapping between private ip address to public ip address on router or firewall on internet gateway.
- Equipment who service nat operation will have nat table for keep record between original and translate ip address.
- NAT 1-to-1 (static nat): source and translate ip address will mapping one-by-one
- NAT 1-to-many (dynamic nat): many source will map to single translate ip address

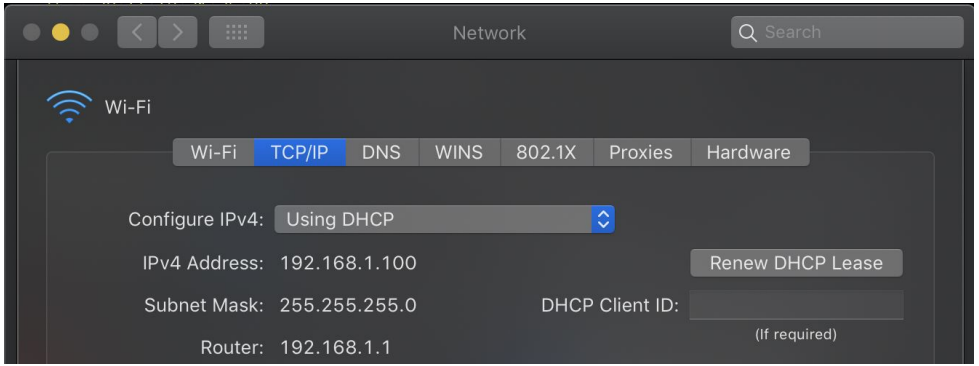
NAT Operation



Ref: https://en.wikipedia.org/wiki/Network_address_translation#/media/File:Bidirectional_NAT-en.svg

Binary math for subnet mask

- **Example**: If your computer had been set ip address like below
  - IP Address: 192.168.1.100
  - Subnet Mask: 255.255.255.0
  - Router/Default Gateway: 192.168.1.1
- As know-how we got before. This should be private ip address of “class c” network that can contain 254 hosts on this network
- But?...How exactly computer know that range of network ?
- How to define which ip address is same network? Which ip address is on different network ?



Binary math for subnet mask

- As computer operate everything with binary base (include ip address)
- So on this point-of-view we can fulfill ability for determined ip address on network segment and host segment by “Subnet Mask”
- Conceptual: Subnet mask will mask for determined part of sub-network by add “1” on bits of network segment and add “0” on bits of hosts segment
- We can generate subnet mask with step below and convert to decimal
- Example: 10.105.30.23/16
  - Decimal: 10 . 105 . 30 . 23
  - Binary: **00001010** . **01101001** . 00011110 . 00010111
  - Subnet: **11111111** . **11111111** . 00000000 . 00000000
  - Decimal: 255 . 255 . 0 . 0

Binary math for subnet mask

- Example: 192.168.1.100/24
  - Decimal: 192 . 168 . 1 . 100
  - Binary: **11000000** . **10101000** . **00000001** . 00011000
  - Subnet: **11111111** . **11111111** . **11111111** . 0000000
  - Decimal: 255 . 255 . 255 . 0
-
- Exzmple: 172.16.89.24/20
  - Decimal: 172 . 16 . 89 . 24
  - Binary: **10101100** . **00010000** . **0101**1001 . 00011000
  - Subnet: **11111111** . **11111111** . **1111**0000 . 0000000
  - Decimal: 255 . 255 . 240 . 0

Binary math for subnet mask

- How to define address range with subnet mask ?
  - For operate with subnet mask. Computer will operate by “AND” operation
    - 1 AND 1 = 1
    - 1 AND 0 = 0
    - 0 AND 1 = 0
    - 0 AND 0 = 0
  - Step 1: Calculate host ip address with subnet mask ⇒ keep result as reference (A: Source NetID)
  - Step 2: Calculate destination ip address with subnet mask ⇒ keep result as reference (B: Dest NetID)
  - Step 3: Compare result A and B
    - If A = B ⇒ source and destination is on same subnet (L2)
    - If A != B ⇒ source and destination is on different subnet. So we will consideration routing process for next step ^^

Binary math for subnet mask

- **Example**:
  - Host ip address: 10.101.200.250/24
  - Destination ip address:
    - Destination 1: 10.101.230.10
    - Destination 2: 10.101.200.30
  - Step 1: Calculate host address

Decimal	10	101	200	250	
Binary	00001010	01100101	11001000	11111010	
Subnet Mask	**11111111**	**11111111**	**11111111**	00000000	(255.255.255.0)
And operator	00001010	01100101	11001000	00000000	(10.101.200.0)

Binary math for subnet mask

- **Example**:
  - Step 2: Calculate destination address
    - Destination 1: 10.101.230.10

Decimal	10	101	230	10	
Binary	00001010	01100101	11100110	00001100	
Subnet Mask	**11111111**	**11111111**	**11111111**	00000000	
AND operator	00001010	01100101	11100110	00000000	(10.101.230.0)

  - Step 3: Compare with source netid

Source NetID	00001010	01100101	11001000	00000000	(10.101.200.0)
Dest NetID	00001010	01100101	11100110	00000000	(10.101.230.0)

    - Result: destination on different network

Binary math for subnet mask

- **Example**:
  - Step 2: Calculate destination address
    - Destination 2: 10.101.200.30

Decimal	10	101	200	30	
Binary	00001010	01100101	11001000	00011110	
Subnet Mask	**11111111**	**11111111**	**11111111**	00000000	
AND operator	00001010	01100101	11001000	00000000	(10.101.200.0)

  - Step 3: Compare with source netid

Source NetID	00001010	01100101	11001000	00000000	(10.101.200.0)
Dest NetID	00001010	01100101	11001000	00000000	(10.101.200.0)

    - Result: destination on same network

Practice 1.3: Define address by subnet mask

- **Ex1**:
  - Host ip address: 172.16.211.129/25
  - Destination ip address:
    - Destination 1: 172.16.211.126
    - Destination 2: 172.16.212.130
  - Step 1: Calculate host address

Decimal	172	16	211	129	
Binary	10101100	00010000	11010011	10000001	
Subnet Mask	**????????**	**????????**	**????????**	**?**???????	(x.x.x.x)
And operator	????????	????????	????????	????????	(x.x.x.x)

Practice 1.3: Define address by subnet mask

- **Ex1**:
  - Host ip address: 172.16.211.129/25
  - Destination ip address:
    - Destination 1: 172.16.211.126
    - Destination 2: 172.16.212.130
  - Step 1: Calculate host address

Decimal	172	16	211	129	
Binary	10101100	00010000	11010011	10000001	
Subnet Mask	**11111111**	**11111111**	**11111111**	**1**0000000	(255.255.255.128)
And operator	10101100	00010000	11010011	10000000	(172.16.211.128)

Practice 1.3: Define address by subnet mask

- **Ex1**:
  - Step 2: Calculate destination address
    - Destination 1: 172.16.211.126

Decimal	172	16	211	126	
Binary	10101100	00010000	11010011	01111110	
Subnet Mask	**11111111**	**11111111**	**11111111**	**1**0000000	
AND operator	????????	????????	????????	????????	(x.x.x.x)

  - Step 3: Compare with source netid

Source NetID	10101100	00010000	11010011	10000000	(x.x.x.x)
Dest NetID	????????	????????	????????	????????	(x.x.x.x)

    - Result: ????

Practice 1.3: Define address by subnet mask

- **Ex1**:
  - Step 2: Calculate destination address
    - Destination 1: 172.16.211.126

Decimal	172	16	211	126
Binary	10101100	00010000	11010011	01111110
Subnet Mask	**11111111**	**11111111**	**11111111**	**1**0000000
AND operator	10101100	00010000	11010011	00000000

(172.16.211.0)

  - Step 3: Compare with source netid

Source NetID	10101100	00010000	11010011	10000000
Dest NetID	10101100	00010000	11010011	00000000

(172.16.211.128)

(172.16.200.0)

    - Result: destination on different network

Practice 1.3: Define address by subnet mask

- **Ex1**:
  - Step 2: Calculate destination address
    - Destination 2: 172.16.212.130

Decimal	172	16	212	130
Binary	10101100	0010000	11010100	10000010
Subnet Mask	**11111111**	**11111111**	**11111111**	**1**0000000
AND operator	????????	????????	????????	????????

(x.x.x.x)

  - Step 3: Compare with source netid

Source NetID	10101100	0010000	11010011	10000000	(x.x.x.x)
Dest NetID	????????	????????	????????	????????	(x.x.x.x)

    - Result: ????

Practice 1.3: Define address by subnet mask

- **Ex1**:
  - Step 2: Calculate destination address
    - Destination 2: 172.16.212.130

Decimal	172	16	212	130
Binary	10101100	0010000	11010100	10000010
Subnet Mask	**11111111**	**11111111**	**11111111**	**1**0000000
AND operator	10101100	0010000	11010100	10000000

(172.16.212.128)

  - Step 3: Compare with source netid

Source NetID	10101100	0010000	11010011	10000000
Dest NetID	10101100	0010000	11010100	10000000

(172.16.211.128)

(172.16.212.128)

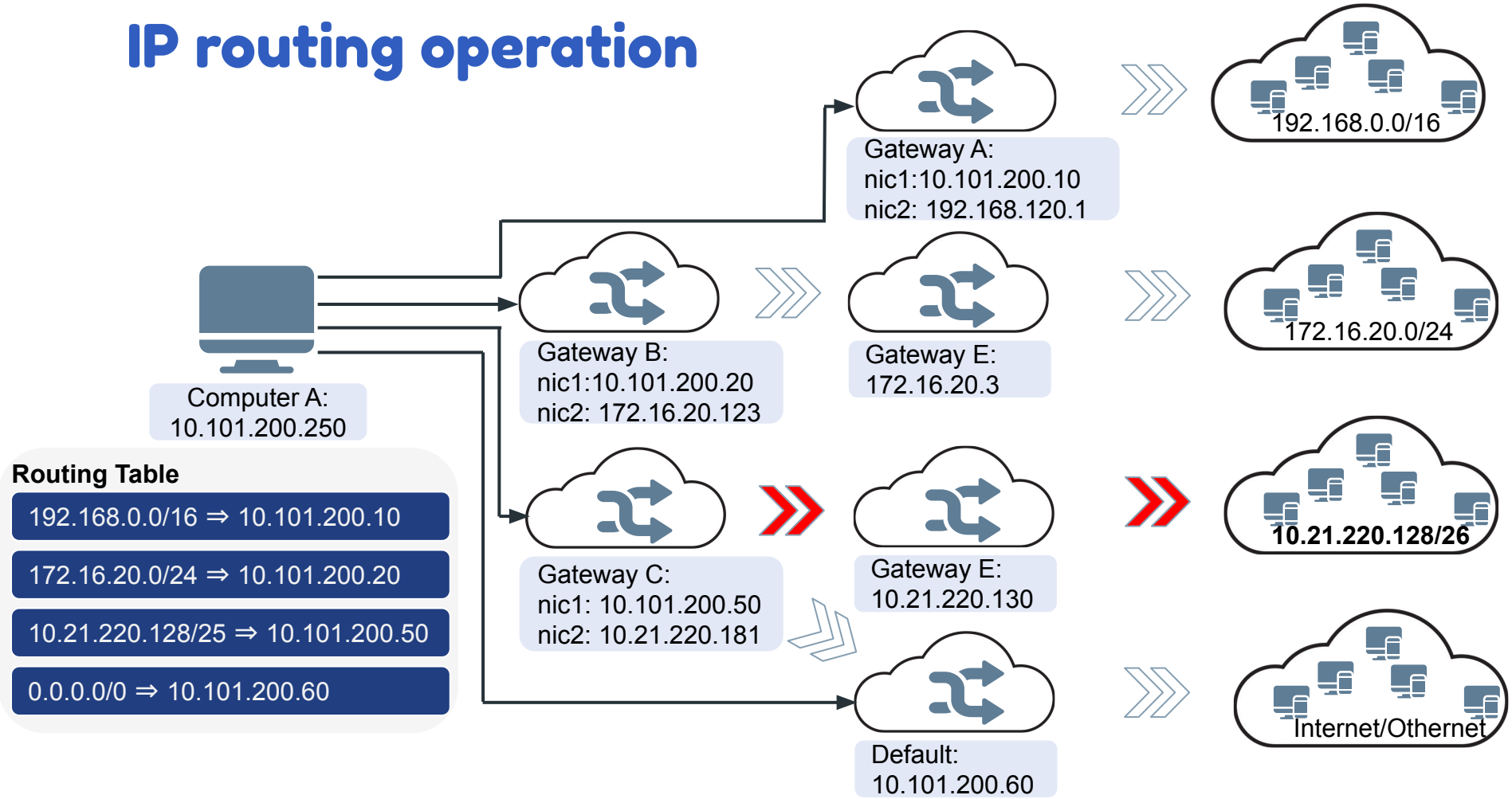
    - Result: destination on same network

IP routing operation

- If computer/l3 found that destination ip address is same network (Aka no route . It will send **ARP packet** and start communication with mac address (L2:Ethernet) directly !!!
- What if it found on different subnet ?...
- Routing operation will take role of this !!!
- Routing table will give direction to computer with routing table
- This process will repeat until packet was sent to destination network
- **Example**: Host ip address: 10.101.200.250/24

Destination	Subnet Mask	Gateway/Next Hop	Interface	Remark
192.168.0.0/16	255.255.0.0	10.101.200.10	Eth1	Static Route
172.16.20.0/24	255.255.255.0	10.101.200.20	Eth1	Static Route
10.21.220.128/25	255.255.255.128	10.101.200.50	Eth1	Static Route
0.0.0.0/0	0.0.0.0	10.101.200.60	Eth1	Default Route

IP routing operation



IPv4 and IPv6

- IPv4 is our favorite as learn today. But it almost exhaust on the world.
- Internet Engineer Task Force (IETF) was recognize for this problem and start to design IPv6 since 1998 and announce to internet standard on 14 july 2017 (RFC 8200, RFC 8201)
- IPv6 was design to compatibility with IPv4 for address, routing and also enhance some feature
    - **Large address space**: IPv6 was extend to 128 bits
    - **Multicasting**: IPv6 have multicast ip address (ff02::1) with rendezvous point addresses for multicast group address
    - **Stateless address autoconfiguration (SLAAC)**: Self generate ip address
    - **IPSec:** Original design for IPv6
    - **Simplified process by router:** Optimized ip packet header
    - **Support extension:** Provide extension header for support extend feature on future

IPv4 and IPv6

Type	IP Address
General	2001:0db8:0000:0000:0000:ff00:0042:8329
Loop back	0000:0000:0000:0000:0000:0000:0000:0001
Private IP	fd00:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx (fd00::/8)

IPv4 and IPv6

- IP packet (IPv4)

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification																Flags				Fragment Offset											
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															
24	192																																
28	224																																
32	256																																

IPv4 and IPv6

- IP packet (IPv6)

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	*Version*				*Traffic Class*								*Flow Label*																			
4	32	*Payload Length*																*Next Header*								*Hop Limit*							
8	64	*Source Address*																															
12	96																																
16	128																																
20	160																																
24	192	*Destination Address*																															
28	224																																
32	256																																
36	288																																

Workshop 1.3 Routing Operation

```
ubuntu@ip-10-21-2-237:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc fq_codel state UP group default qlen 1000
    link/ether 06:11:cc:15:31:04 brd ff:ff:ff:ff:ff:ff
    inet 10.21.2.237/24 brd 10.21.2.255 scope global dynamic eth0
       valid_lft 1912sec preferred_lft 1912sec
    inet6 fe80::411:ccff:fe15:3104/64 scope link
       valid_lft forever preferred_lft forever
ubuntu@ip-10-21-2-237:~$ ip r
default via 10.21.2.1 dev eth0 proto dhcp src 10.21.2.237 metric 100
10.21.2.0/24 dev eth0 proto kernel scope link src 10.21.2.237
10.21.2.1 dev eth0 proto dhcp scope link src 10.21.2.237 metric 100
ubuntu@ip-10-21-2-237:~$
ubuntu@ip-10-21-2-237:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.21.2.1       0.0.0.0         UG        0 0          0 eth0
10.21.2.0       0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.21.2.1       0.0.0.0         255.255.255.255 UH        0 0          0 eth0
```

```
ubuntu@ip-10-21-2-237:~$ dig www.google.com

; <<>> DiG 9.11.3-1ubuntu1.13-Ubuntu <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2256
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;www.google.com.                        IN      A

;; ANSWER SECTION:
www.google.com.         65      IN      A       74.125.24.147
www.google.com.         65      IN      A       74.125.24.106
www.google.com.         65      IN      A       74.125.24.105
www.google.com.         65      IN      A       74.125.24.104
www.google.com.         65      IN      A       74.125.24.103
www.google.com.         65      IN      A       74.125.24.99

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Sat Dec 26 16:16:45 UTC 2020
;; MSG SIZE  rcvd: 139

ubuntu@ip-10-21-2-237:~$ traceroute -w 5 -m 100 74.125.24.147
traceroute to 74.125.24.147 (74.125.24.147), 100 hops max
  1   18.141.171.9  12.050ms  71.968ms  9.996ms
  2   100.65.32.96  2.193ms  8.125ms  8.270ms
  3   100.66.16.252  108.625ms  0.847ms  8.742ms
  4   100.66.18.238  21.185ms  21.943ms  21.835ms
  5   100.66.6.235  18.545ms  11.734ms  21.486ms
  6   100.66.4.245  11.480ms  22.008ms  21.908ms
  7   100.65.8.97  0.414ms  0.432ms  0.412ms
  8   203.83.223.20  1.759ms  1.249ms  1.254ms
  9   52.93.8.94  2.571ms  3.889ms  *
 10   52.93.10.135  1.244ms  1.207ms  1.160ms
 11   99.82.177.13  2.104ms  2.100ms  2.365ms
 12   *  *  *
 13   108.170.254.225  2.742ms  2.531ms  2.607ms
 14   108.170.254.227  8.540ms  9.353ms  10.754ms
 15   216.239.49.224  3.386ms  3.505ms  3.421ms
 16   74.125.252.254  3.366ms  3.463ms  3.358ms
 17   *  *  *
 18   *  *  *
 19   *  *  *
 20   *  *  *
 21   *  *  *
 22   *  *  *
 23   *  *  *
 24   *  *  *
 25   *  *  *
 26   *  *  *
 27   74.125.24.147  2.094ms  2.038ms  2.042ms
ubuntu@ip-10-21-2-237:~$
```

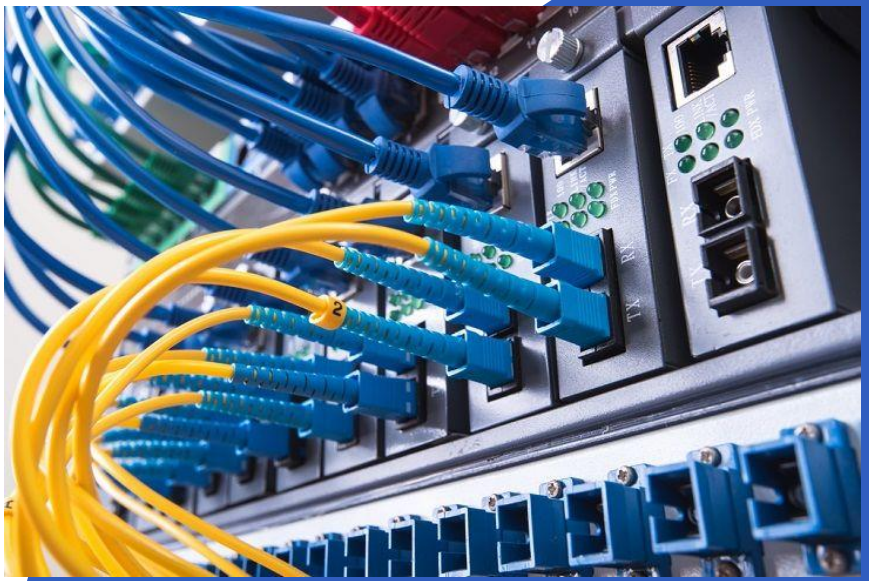
Agenda (Recap)

- **Day1**
- Network and Communication Concept
  - Concept of communication
  - Protocol on referencing
  - Conductor/Carrier
- Walkthrough OSI 7 Layer
  - What is OSI module
  - Dataflow and encapsulation concept
  - L1 - L7 description
  - Advance concept:
    - VLAN (RFC 5517/IEEE802.1Q)
    - VXLAN (RFC7348)
- TCP/IP Architecture:
  - Introduction to TCP/IP
  - Reflect architecture between TCP/IP and OSI

Agenda (Recap)

- **Day1 (Continue)**
- IP Address and routing (OSI: L3, TCP/IP: L2)
  - Mathematic and IP address overview
  - IP Address structure, class (A,B,C) and CIDR
  - IP Private Addresses
  - NAT operation
  - Binary math for subnet mask
  - IP routing operation
  - ARP operation
  - IPv4 vs IPv6

TCP/UDP flow control



TCP/UDP Flow Control

- Both TCP/UDP is on layer 4 (TCP/IP) and layer 4 - 5 (OSI Model)
- It maintain end-to-end connection for reliable, consistency and session management on this layer
- Connection on this level was categorized like below
  - Unicast: One-to-One connection
  - Multicast: One-to-Multiple connection
    - 224.0.0.0 to 239.255.255.255 (L3)
    - 01-00-5E-xx-xx-xx (L2)
  - Broadcast: One-to-All subnet/domain network
    - 255.255.255.0 (L3)
    - ff-ff-ff-ff-ff-ff (L2)

Connection Oriented / Connectionless

- Connection Oriented (StateFul)
- Connectionless (Stateless)

TCP Flow Operate

- **TCP (Transmission Control Protocol) (RFC 793: IPv4, RFC2460: IPv6)**
  - **Born to be reliable:** Control transmission with reliable, ordered, error-checked and end-to-end communication (Loss performance for communication and re-transmission around 20 - 30%)
  - **Connection-Oriented**: Between client-server need to established connection before start communication (Control by sequence number)
  - **Sliding windows protocol:** For control data sent and receive on optimize size between client-server
  - **Checksum:** Control error on tcp will use crc checksum for detect complete

TCP Flow Operate

- TCP packet (IPv4)

Bit offset	0–3	4–7	8–15	16–31
0	Source address			
32	Destination address			
64	Zeros		Protocol	TCP length
96	Source port			Destination port
128	Sequence number			
160	Acknowledgement number			
192	Data offset	Reserved	Flags	Window
224	Checksum			Urgent pointer
256	Options (optional)			
256/288+	Data			

TCP Flow Operate

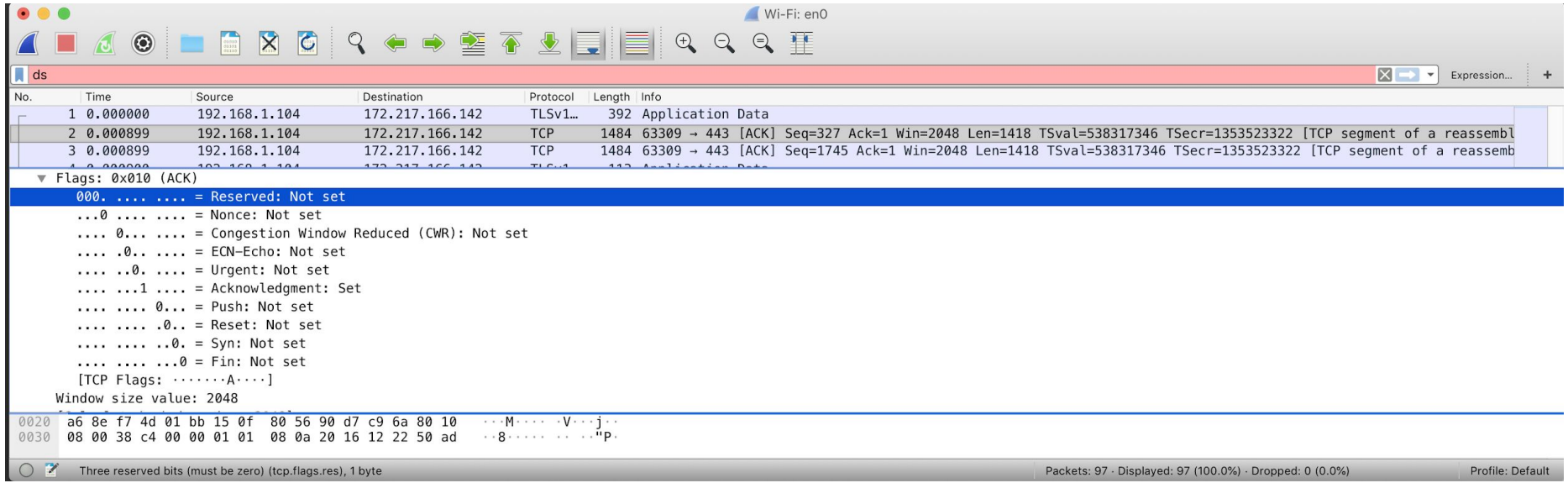
- TCP packet (IPv6)

Bit offset	0–7	8–15	16–23	24–31
0	Source address			
32				
64				
96				
128	Destination address			
160				
192				
224				
256	TCP length			
288	Zeros			Next header = Protocol
320	Source port		Destination port	
352	Sequence number			
384	Acknowledgement number			
416	Data offset	Reserved	Flags	Window
448	Checksum		Urgent pointer	
480	Options (optional)			
480/512+	Data			

TCP flag control

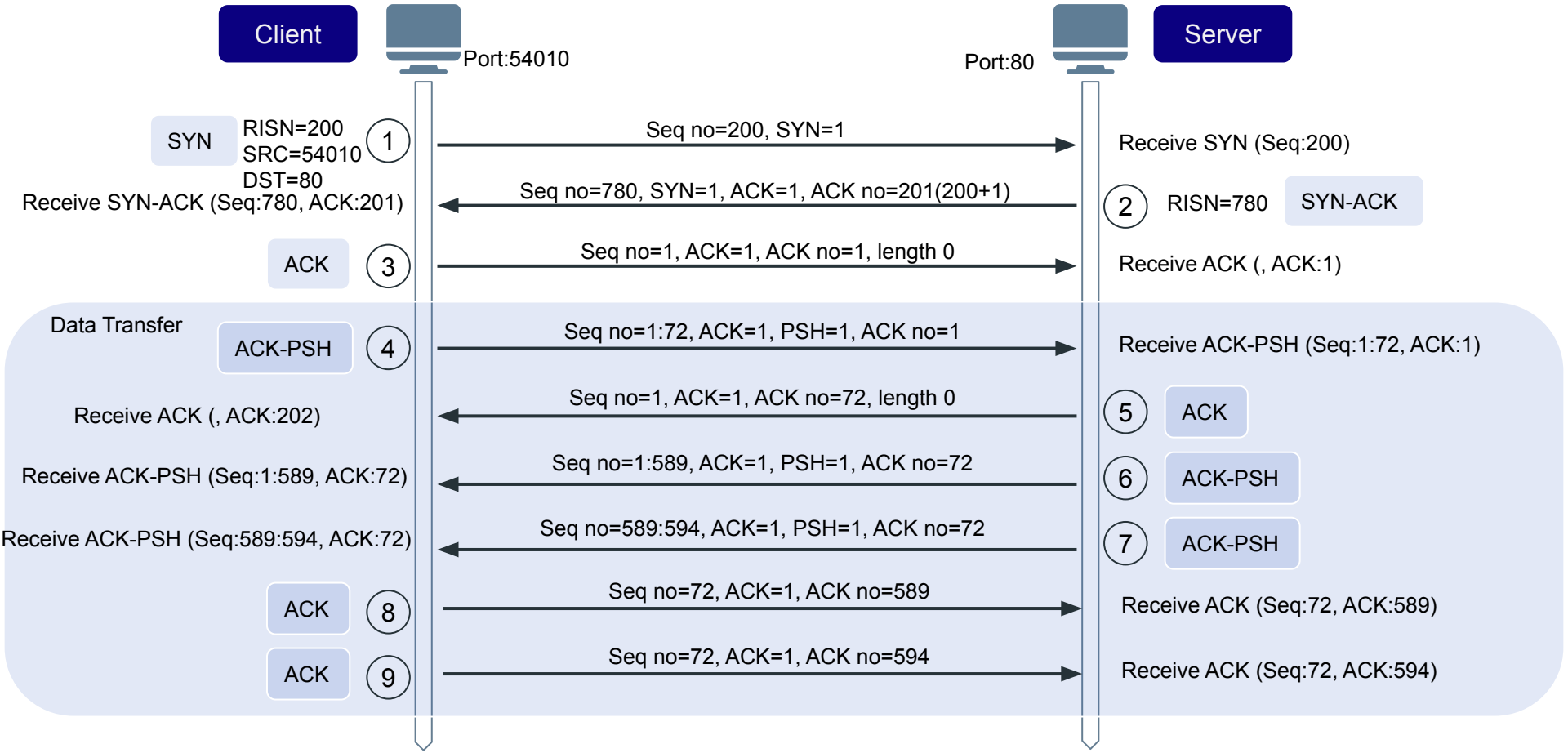
- For fully control all traffic send/receive via TCP need to establish connection between sender-receiver (aka three-way handshake)
- When establish tcp with this step. Both side need to reserve port for establish connection (source port, destination port)
- This process will control any connection for inform both side about happening on connection via **TCP flag (12 bits with reserve 3 bits)**
  - **NS**: ECN-nonce for protect
  - **CWR:** Congestion windows reduce
  - **ECE**: ECN-Echo (1=ECN on, 0=ECN off)
  - **URG**: Urgent data need to sent
  - **ACK**: Acknowledge of all previous state of SYN
  - **PSH**: Ask to push data to receiver (sender empty write buffer etc)
  - **RST**: Reset tcp session when establish
  - **SYN**: Establish tcp session
  - **FIN**: Gracefully terminate a tcp session

TCP flag control

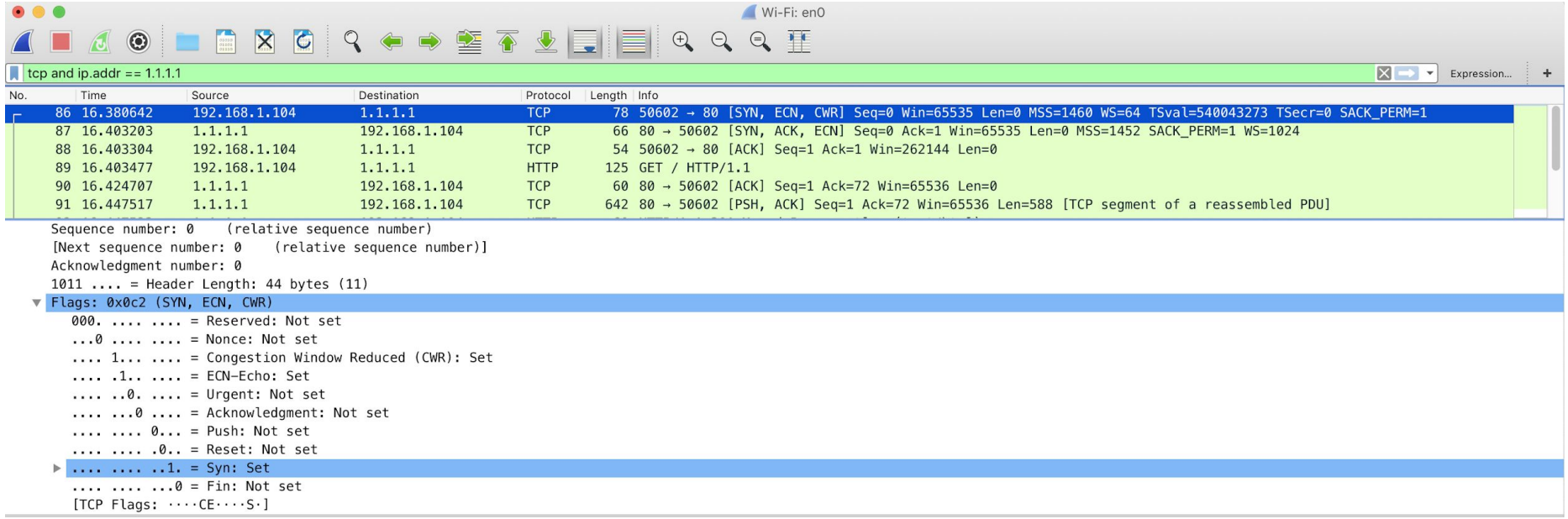


TCP three-way handshake

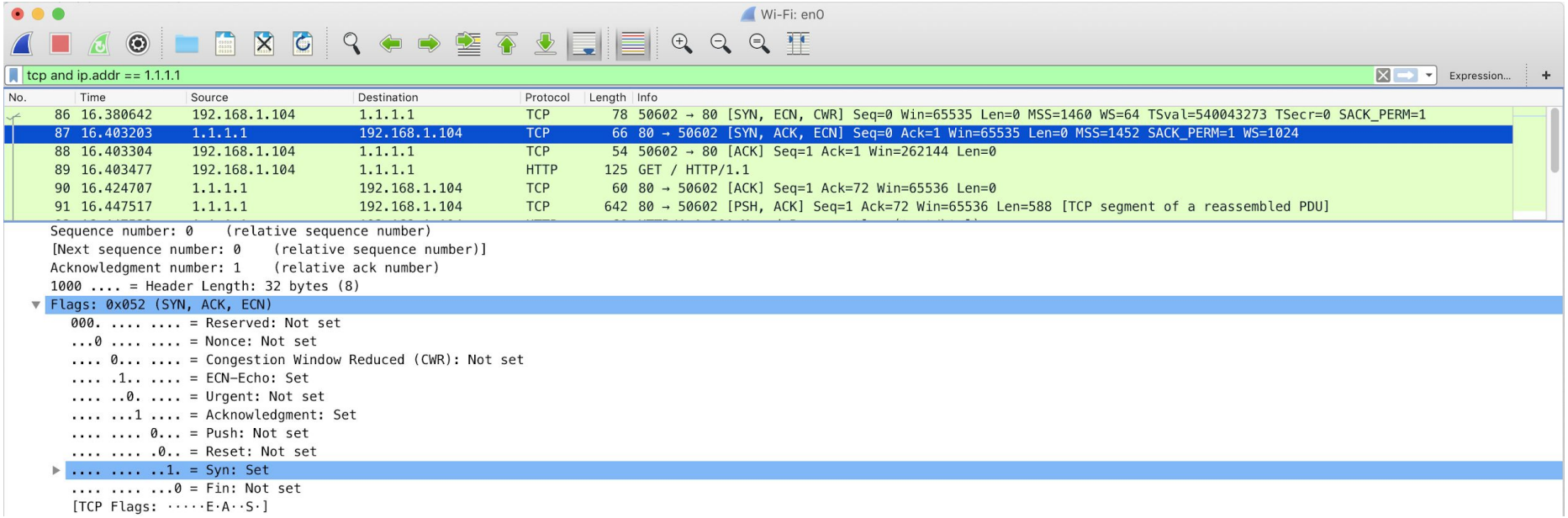
RISN=Random Initial Sequence Number
SRC=Source Port
DST=Destination Port



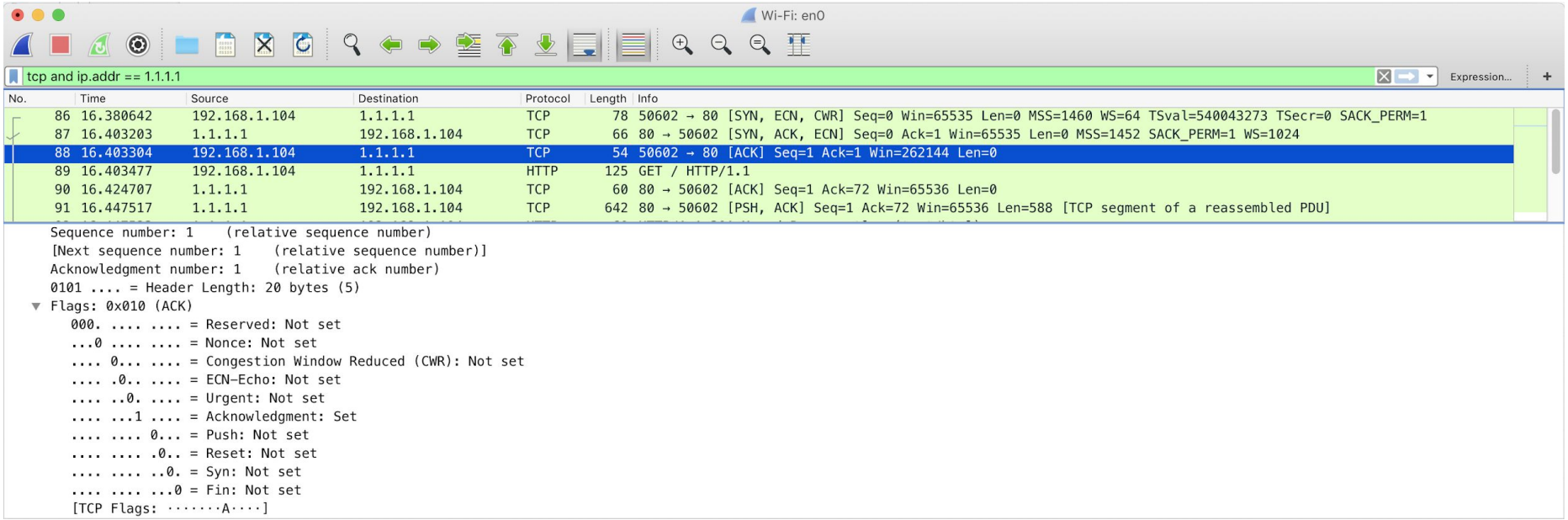
TCP three-way handshake



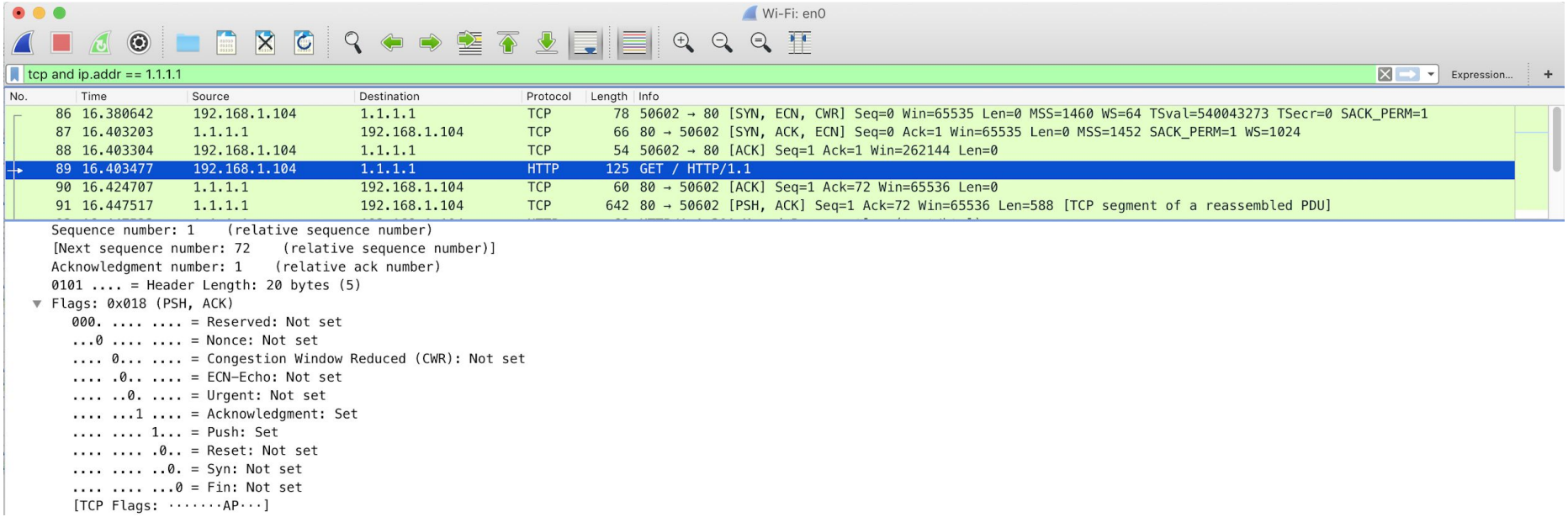
TCP three-way handshake



TCP three-way handshake



TCP three-way handshake

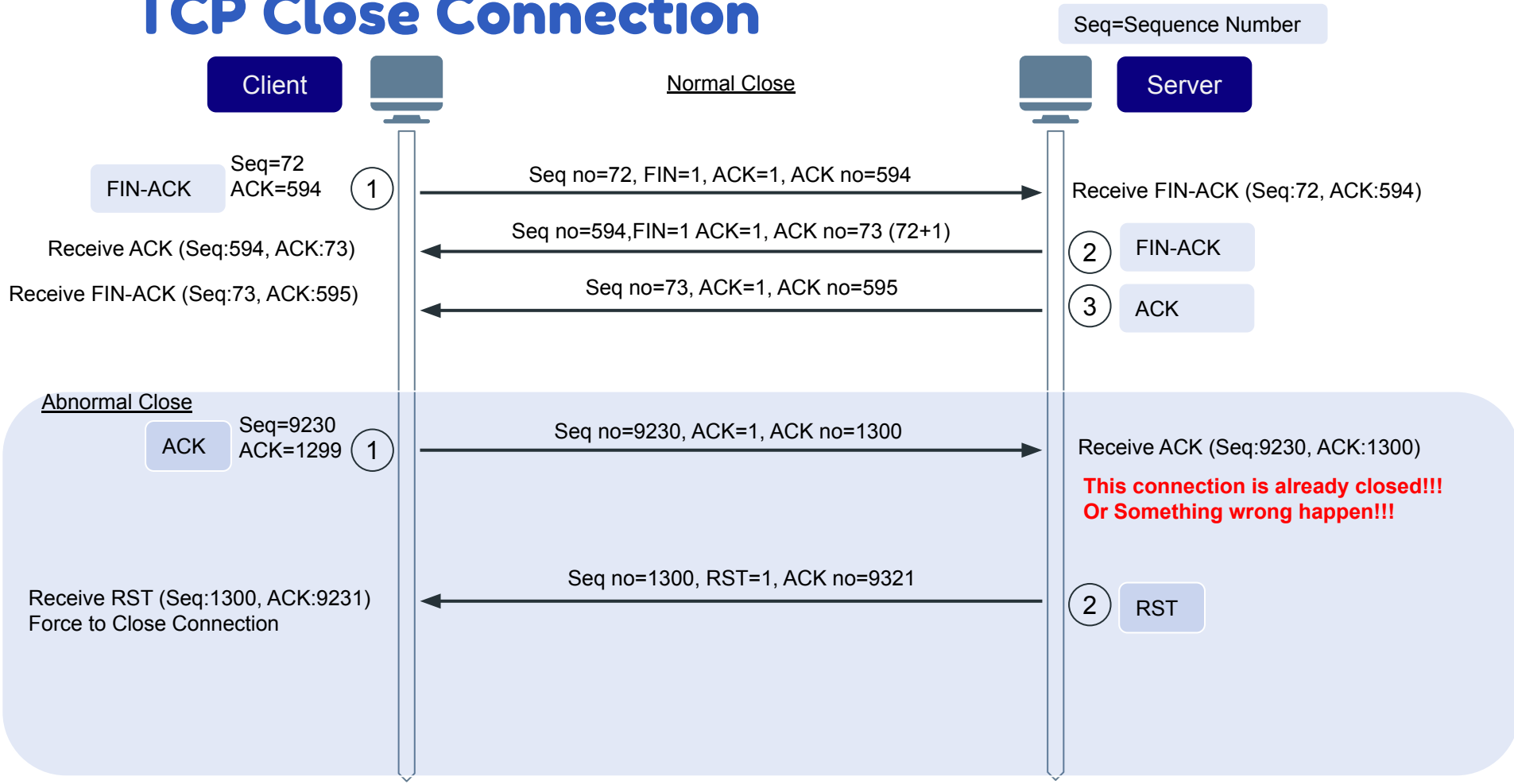


TCP three-way handshake

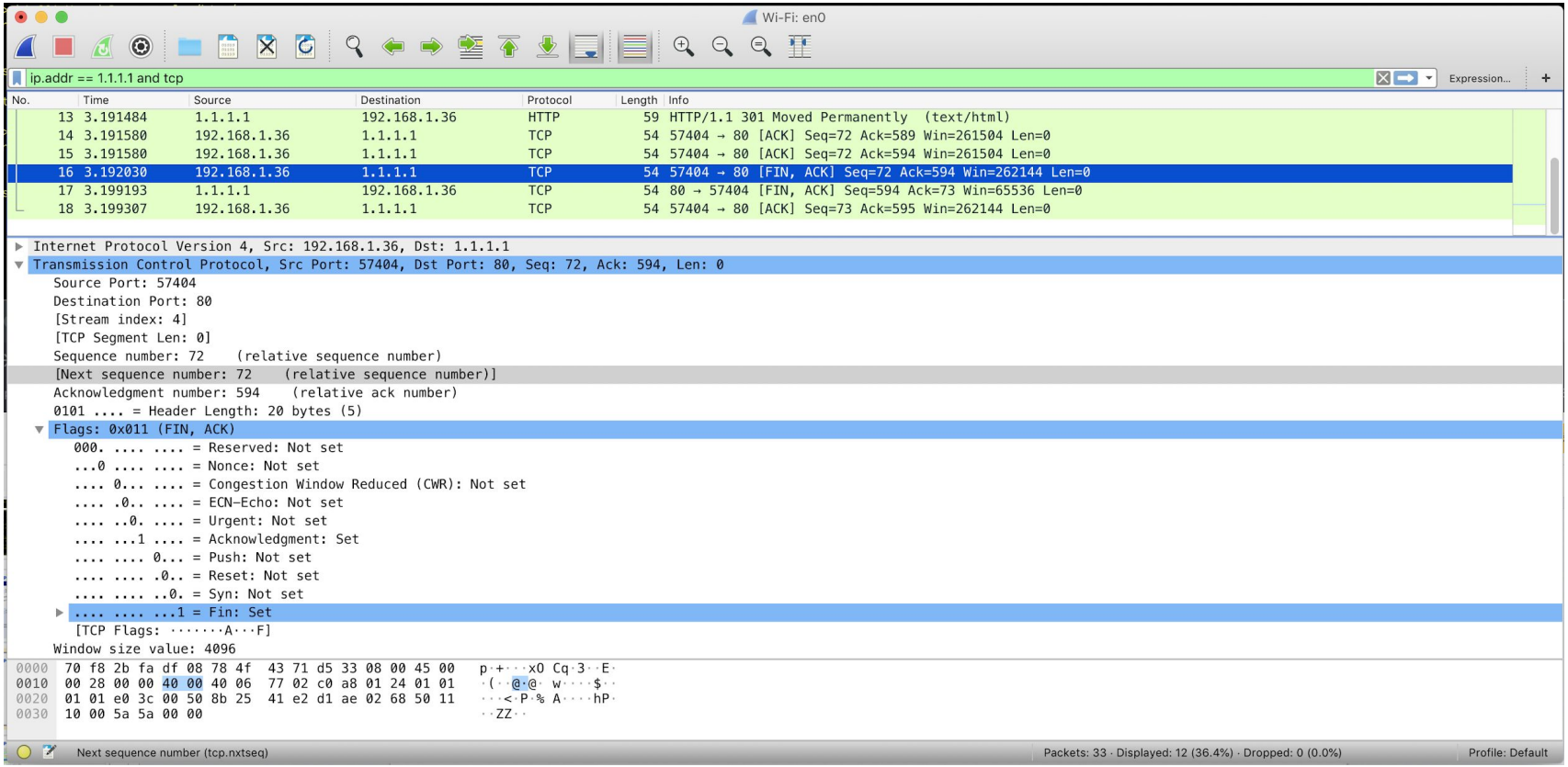
```
praparns-MacBook-Pro:~ praparn$ sudo tcpdump -enni en0 host 1.1.1.1 and tcp port 80
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on en0, link-type EN10MB (Ethernet), capture size 262144 bytes
23:50:11.962427 78:4f:43:71:d5:33 > 74:c9:a3:f9:b1:55, ethertype IPv4 (0x0800), length 78: 192.168.1.104.51784 > 1.1.1.1.80: Flags [SEW], seq 2170944254, win 65535, options [mss 1460,nop,wscale 6,nop,nop,TS val 544144878 ecr 0,sackOK,eol], length
0
23:50:11.985229 74:c9:a3:f9:b1:55 > 78:4f:43:71:d5:33, ethertype IPv4 (0x0800), length 66: 1.1.1.1.80 > 192.168.1.104.51784: Flags [S.E], seq 3153008468, ack 2170944255, win 65535, options [mss 1452,nop,nop,sackOK,nop,wscale 10], length 0
23:50:11.985328 78:4f:43:71:d5:33 > 74:c9:a3:f9:b1:55, ethertype IPv4 (0x0800), length 54: 192.168.1.104.51784 > 1.1.1.1.80: Flags [.], ack 1, win 4096, length 0
23:50:11.985505 78:4f:43:71:d5:33 > 74:c9:a3:f9:b1:55, ethertype IPv4 (0x0800), length 125: 192.168.1.104.51784 > 1.1.1.1.80: Flags [P.], seq 1:72, ack 1, win 4096, length 71: HTTP: GET / HTTP/1.1
```

```
praparns-MacBook-Pro:~ praparn$ curl http://1.1.1.1 -v
* Rebuilt URL to: http://1.1.1.1/
*   Trying 1.1.1.1...
* TCP_NODELAY set
* Connected to 1.1.1.1 (1.1.1.1) port 80 (#0)
> GET / HTTP/1.1
> Host: 1.1.1.1
> User-Agent: curl/7.54.0
> Accept: */*
>
< HTTP/1.1 301 Moved Permanently
< Date: Wed, 09 Dec 2020 17:12:48 GMT
< Content-Type: text/html
< Transfer-Encoding: chunked
< Connection: keep-alive
< Location: https://1.1.1.1/
< Served-In-Seconds: 0.000
< CF-Cache-Status: HIT
< Age: 1182
< Expires: Wed, 09 Dec 2020 21:12:48 GMT
< Cache-Control: public, max-age=14400
< cf-request-id: 06ea16d58000005acee0216000000001
< Server: cloudflare
< CF-RAY: 5ff05a68ce6e5ace-BKK
<
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
<hr><center>cloudflare-lb</center>
</body>
</html>
* Connection #0 to host 1.1.1.1 left intact
praparns-MacBook-Pro:~ praparn$
```

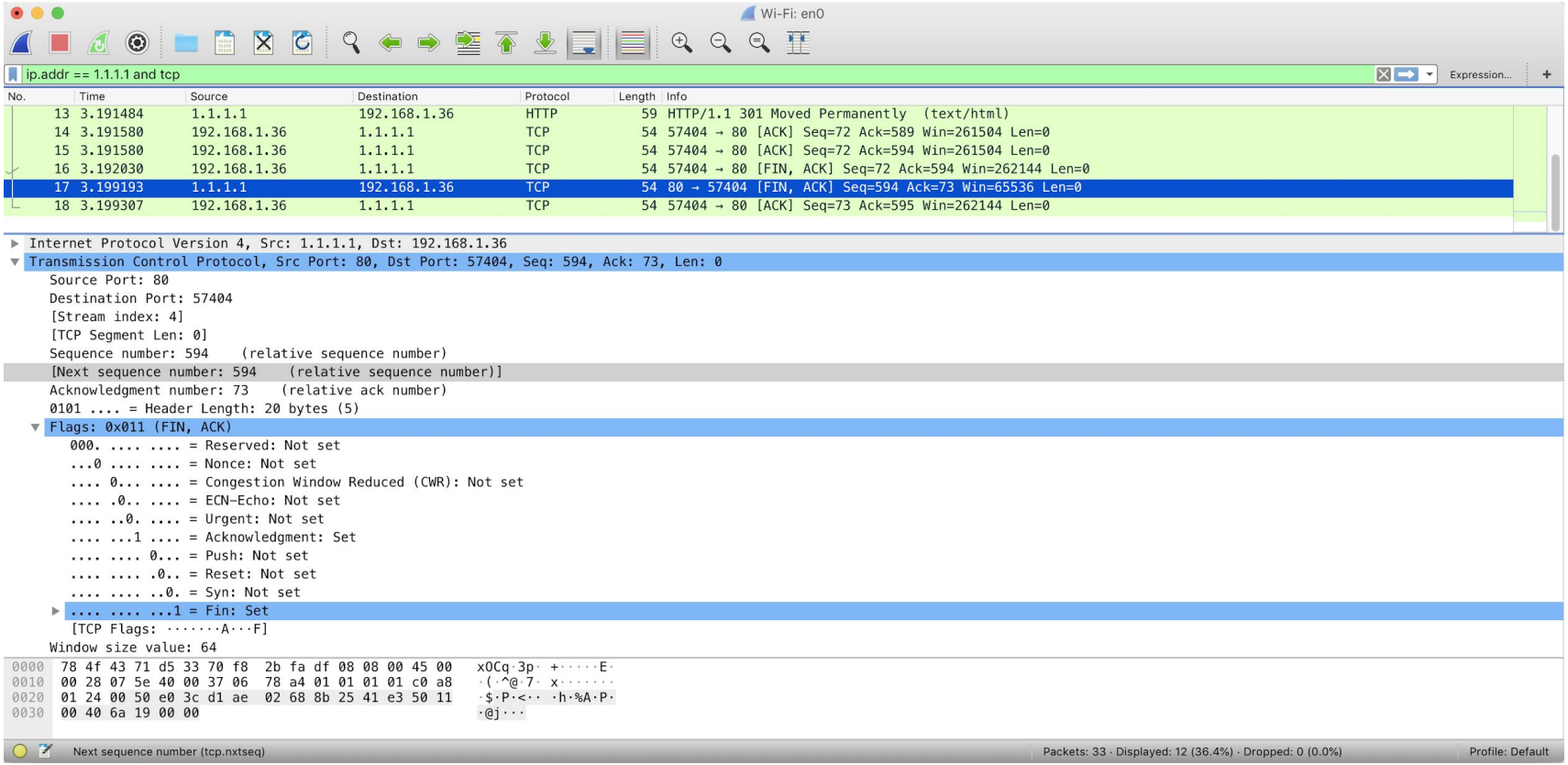
TCP Close Connection



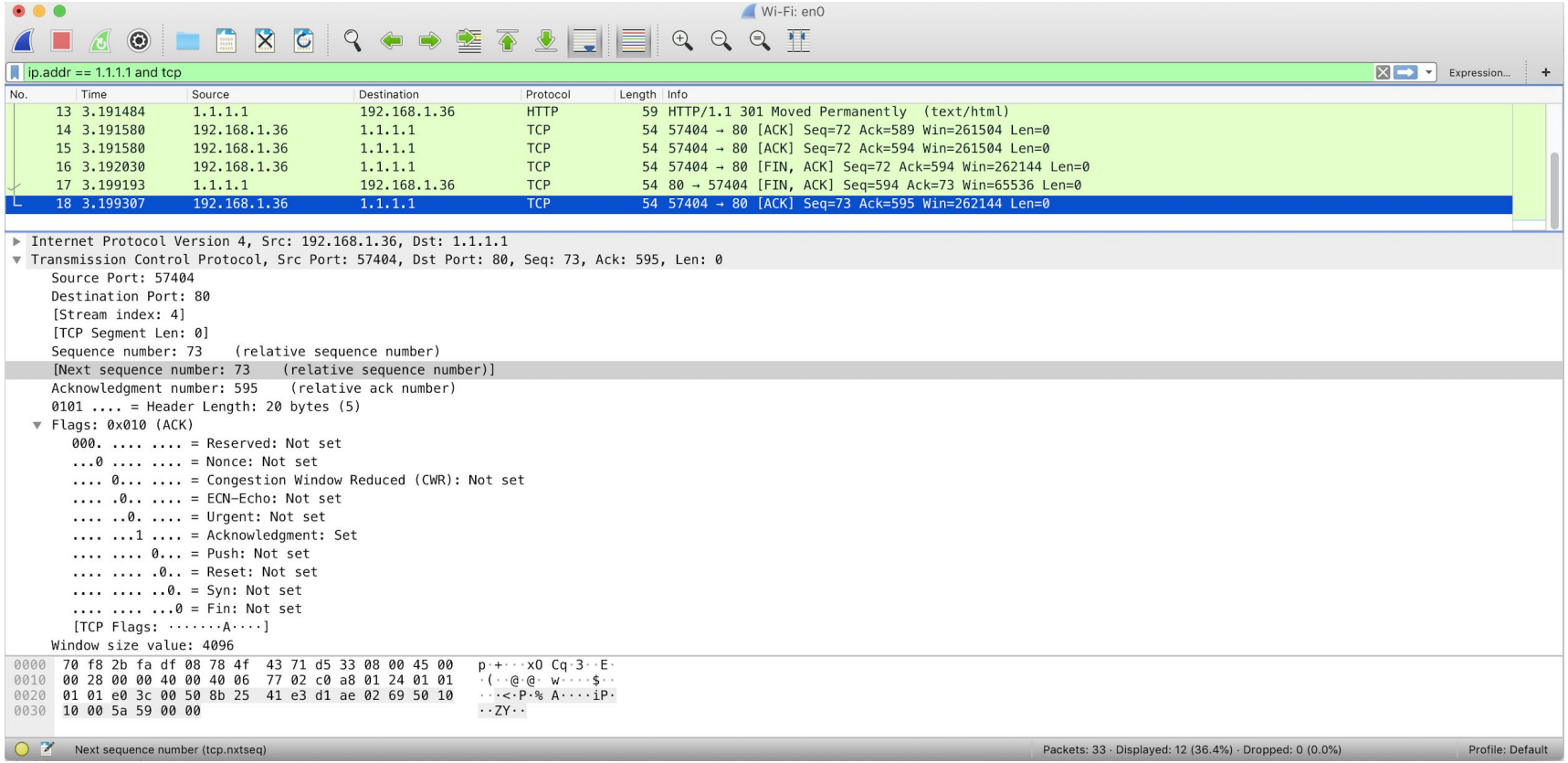
TCP Close Connection



TCP Close Connection



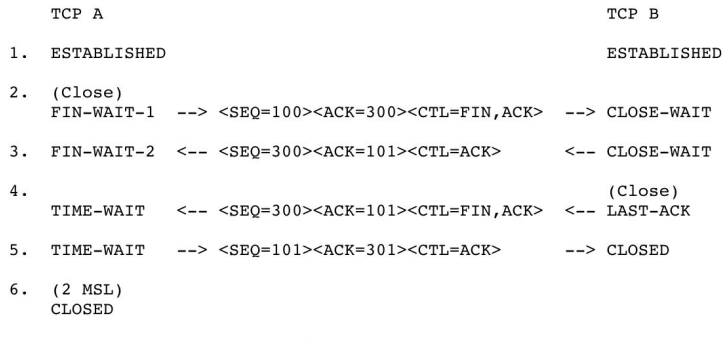
TCP Close Connection



TCP Close Connection

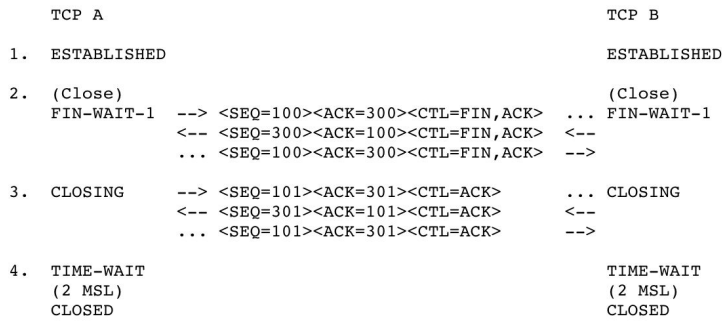
```
17:31:21.405244 06:11:cc:15:31:04 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 54: 10.21.2.237.51114 > 1.1.1.1.80: Flags [F.], seq 72, ack 594, win 487, length 0
        0x0000:  4500 0028 fbfc 4000 4006 2fd0 0a15 02ed  E..(..@.@./.....
        0x0010:  0101 0101 c7aa 0050 0e31 567a 63c5 9cef  .......P.1Vzc...
        0x0020:  5011 01e7 0f1e 0000                      P.......
17:31:21.407757 06:4d:1e:bf:fe:8e > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 54: 1.1.1.1.80 > 10.21.2.237.51114: Flags [F.], seq 594, ack 73, win 64, length 0
        0x0000:  4500 0028 1b82 4000 3306 1d4b 0101 0101  E..(..@.3..K....
        0x0010:  0a15 02ed 0050 c7aa 63c5 9cef 0e31 567b  .....P..c....1V{
        0x0020:  5011 0040 7334 0000                      P..@s4..
17:31:21.407779 06:11:cc:15:31:04 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 54: 10.21.2.237.51114 > 1.1.1.1.80: Flags [.], ack 595, win 487, length 0
        0x0000:  4500 0028 fbfd 4000 4006 2fcf 0a15 02ed  E..(..@.@./.....
        0x0010:  0101 0101 c7aa 0050 0e31 567b 63c5 9cf0  .......P.1V{c...
        0x0020:  5010 01e7 0f1e 0000                      P.......
^C
12 packets captured
12 packets received by filter
0 packets dropped by kernel
ubuntu@ip-10-21-2-237:~$
```

TCP Close Connection



Normal Close Sequence

Figure 13.



Simultaneous Close Sequence

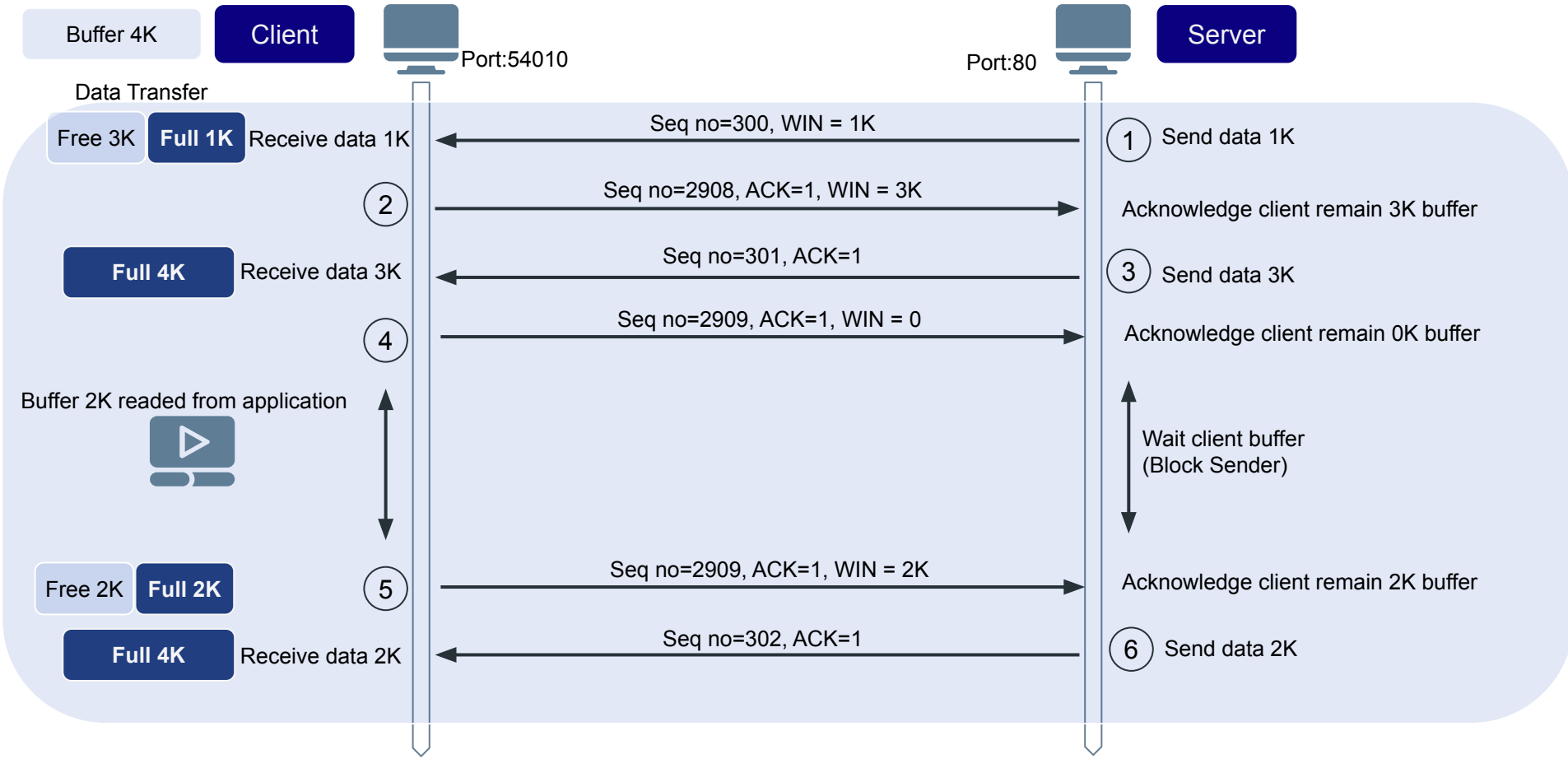
Figure 14.

Ref: https://tools.ietf.org/html/rfc793#section-3.5

TCP Sliding Windows

- After establish state on tcp connection. Sender and receiver will start to send/receive data
- For make sure all that sender and receiver have buffer enough for support communication. TCP will have sliding windows for inform both side about buffer for send/receive that will contain on header “windows”
- When receiver act with window = 0. Sender need to wait until receiver send act with free window
- With this method sender and receiver will ensure that all data never miss from buffer

TCP Sliding Windows



Workshop 1.4 TCP Flow Control

```
ubuntu@ip-10-21-2-237:~$ sudo tcpdump -enni eth0 host 1.1.1.1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
16:52:01.388870 06:11:cc:15:31:04 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 74: 10.21.2.237.51098 > 1.1.1.1.80: Flags [S], seq 2220545583, win 62727, options [mss 8961,sackOK,TS val 3950682488 ecr 0,nop,wscale 7], length 0
16:52:01.390609 06:4d:1e:bf:fe:8e > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 66: 1.1.1.1.80 > 10.21.2.237.51098: Flags [S.], seq 3743002766, ack 2220545584, win 65535, options [mss 1460,nop,nop,sackOK,nop,wscale 10], length 0
16:52:01.390631 06:11:cc:15:31:04 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 54: 10.21.2.237.51098 > 1.1.1.1.80: Flags [.], ack 1, win 491, length 0
16:52:01.390952 06:11:cc:15:31:04 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 125: 10.21.2.237.51098 > 1.1.1.1.80: Flags [P.], seq 1:72, ack 1, win 491, length 71: HTTP: GET / HTTP/1.1
16:52:01.392654 06:4d:1e:bf:fe:8e > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 54: 1.1.1.1.80 > 10.21.2.237.51098: Flags [.], ack 72, win 64, length 0
16:52:01.403265 06:4d:1e:bf:fe:8e > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 642: 1.1.1.1.80 > 10.21.2.237.51098: Flags [P.], seq 1:589, ack 72, win 64, length 588: HTTP: HTTP/1.1 301 Moved Permanently
16:52:01.403265 06:4d:1e:bf:fe:8e > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 59: 1.1.1.1.80 > 10.21.2.237.51098: Flags [P.], seq 589:594, ack 72, win 64, length 5: HTTP
16:52:01.403295 06:11:cc:15:31:04 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 54: 10.21.2.237.51098 > 1.1.1.1.80: Flags [.], ack 589, win 487, length 0
16:52:01.403304 06:11:cc:15:31:04 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 54: 10.21.2.237.51098 > 1.1.1.1.80: Flags [.], ack 594, win 487, length 0
16:52:01.404950 06:11:cc:15:31:04 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 54: 10.21.2.237.51098 > 1.1.1.1.80: Flags [F.], seq 72, ack 594, win 487, length 0
16:52:01.406877 06:4d:1e:bf:fe:8e > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 54: 1.1.1.1.80 > 10.21.2.237.51098: Flags [F.], seq 594, ack 73, win 64, length 0
16:52:01.406893 06:11:cc:15:31:04 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 54: 10.21.2.237.51098 > 1.1.1.1.80: Flags [.], ack 595, win 487, length 0
^C
12 packets captured
12 packets received by filter
0 packets dropped by kernel
ubuntu@ip-10-21-2-237:~$
```

```
ubuntu@ip-10-21-2-237:~$ sudo tcpdump -X -enni eth0 host 1.1.1.1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
16:56:48.562471 06:11:cc:15:31:04 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 74: 10.21.2.237.51104 > 1.1.1.1.80: Flags [S], seq 668182636, win 62727, options [mss 8961,sackOK,TS val 3950969661 ecr 0,nop,wscale 7], length 0
        0x0000:  4500 003c 7690 4000 4006 b528 0a15 02ed  E..<v.@.@..(....
        0x0010:  0101 0101 c7a0 0050 27d3 a86c 0000 0000  .......P'..l....
        0x0020:  a002 f507 0f32 0000 0204 2301 0402 080a  .....2....#.....
        0x0030:  eb7f 033d 0000 0000 0103 0307            ...=........
16:56:48.564290 06:4d:1e:bf:fe:8e > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 66: 1.1.1.1.80 > 10.21.2.237.51104: Flags [S.], seq 4164737773, ack 668182637, win 65535, options [mss 1460,nop,nop,sackOK,nop,wscale 10], length 0
        0x0000:  4500 0034 0000 4000 3306 38c1 0101 0101  E..4..@.3.8.....
        0x0010:  0a15 02ed 0050 c7a0 f83c daed 27d3 a86d  .....P...<..'..m
        0x0020:  8012 ffff f49e 0000 0204 05b4 0101 0402  ................
        0x0030:  0103 030a                                ....
16:56:48.564308 06:11:cc:15:31:04 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 54: 10.21.2.237.51104 > 1.1.1.1.80: Flags [.], ack 1, win 491, length 0
        0x0000:  4500 0028 7691 4000 4006 b53b 0a15 02ed  E..(v.@.@..;....
        0x0010:  0101 0101 c7a0 0050 27d3 a86d f83c daee  .......P'..m.<..
        0x0020:  5010 01eb 0f1e 0000                      P.......
16:56:48.564721 06:11:cc:15:31:04 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 125: 10.21.2.237.51104 > 1.1.1.1.80: Flags [P.], seq 1:72, ack 1, win 491, length 71: HTTP: GET / HTTP/1.1
        0x0000:  4500 006f 7692 4000 4006 b4f3 0a15 02ed  E..ov.@.@.......
        0x0010:  0101 0101 c7a0 0050 27d3 a86d f83c daee  .......P'..m.<..
        0x0020:  5018 01eb 0f65 0000 4745 5420 2f20 4854  P....e..GET./.HT
        0x0030:  5450 2f31 2e31 0d0a 486f 7374 3a20 312e  TP/1.1..Host:.1.
        0x0040:  312e 312e 310d 0a55 7365 722d 4167 656e  1.1.1..User-Agen
        0x0050:  743a 2063 7572 6c2f 372e 3538 2e30 0d0a  t:.curl/7.58.0..
        0x0060:  4163 6365 7074 3a20 2a2f 2a0d 0a0d 0a    Accept:.*/*....
16:56:48.566405 06:4d:1e:bf:fe:8e > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 54: 1.1.1.1.80 > 10.21.2.237.51104: Flags [.], ack 72, win 64, length 0
        0x0000:  4500 0028 08f8 4000 3306 2fd5 0101 0101  E..(..@.3./.....
        0x0010:  0a15 02ed 0050 c7a0 f83c daee 27d3 a8b4  .....P...<..'...
        0x0020:  5010 0040 34ed 0000                      P..@4...
16:56:48.573691 06:4d:1e:bf:fe:8e > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 642: 1.1.1.1.80 > 10.21.2.237.51104: Flags [P.], seq 1:589, ack 72, win 64, length 588: HTTP: HTTP/1.1 301 Moved Permanently
```

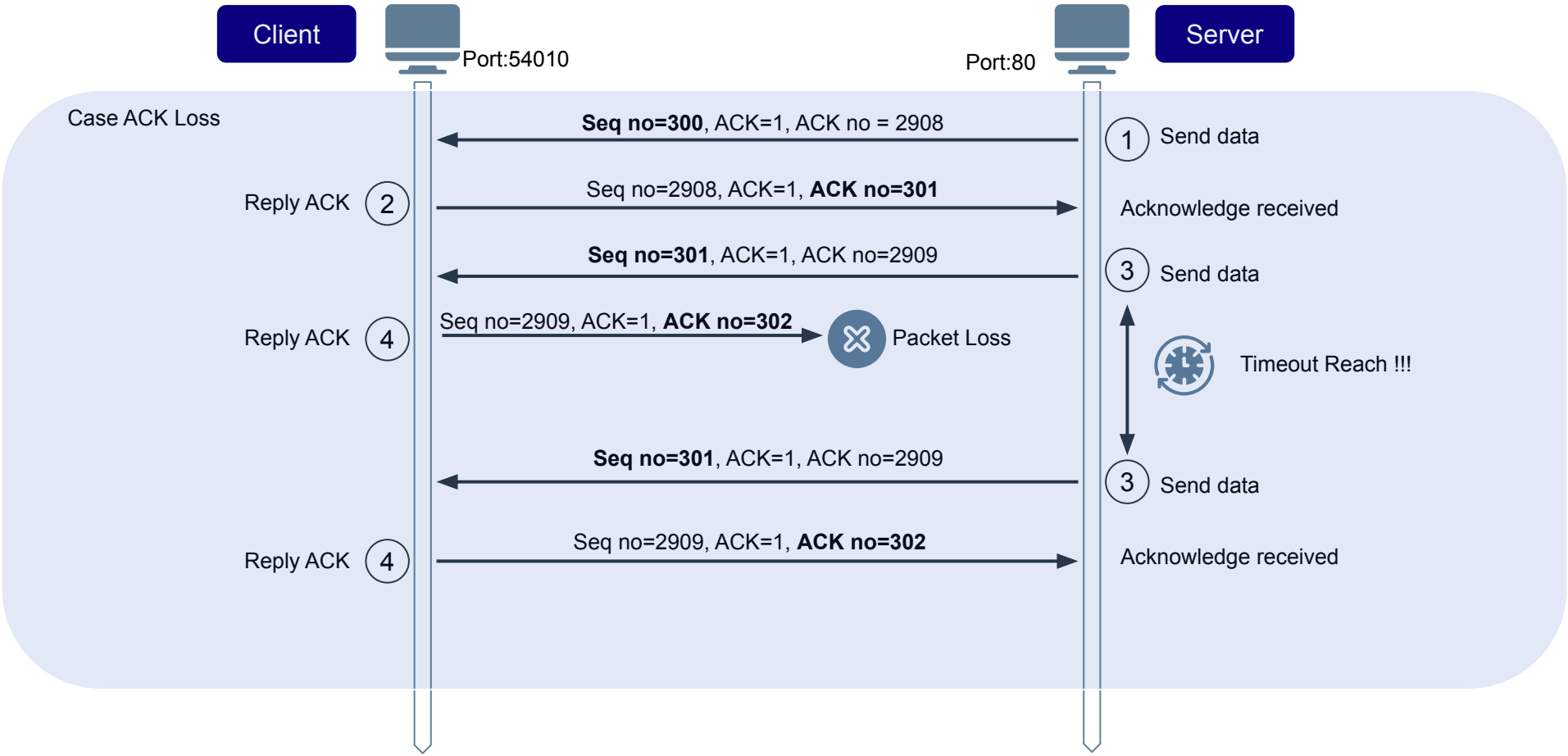
TCP Timeout and Retransmission

- Any connection between sender and receiver is not 100% successful
- Ton of unexpected event may occur
  - Inconsistency media (cross-talk, white-noise, high latency etc)
  - L2, L3 equipment failure
  - L4 - L7 equipment's session timeout
  - etc
- From incident above bring incomplete/loss data between sender and receiver
  - Packet data from sender not arrive to receiver
  - Packet reply (ACK) from receiver not arrive to sender
- In case of incomplete. TCP have method for checking completeness of packet by calculate "checksum"
  - Checksum = ip pseudo + tcp header + tcp body
- Receiver that detect tcp packet that fail for compare checksum will abondorn and wait for retransmission again from sender

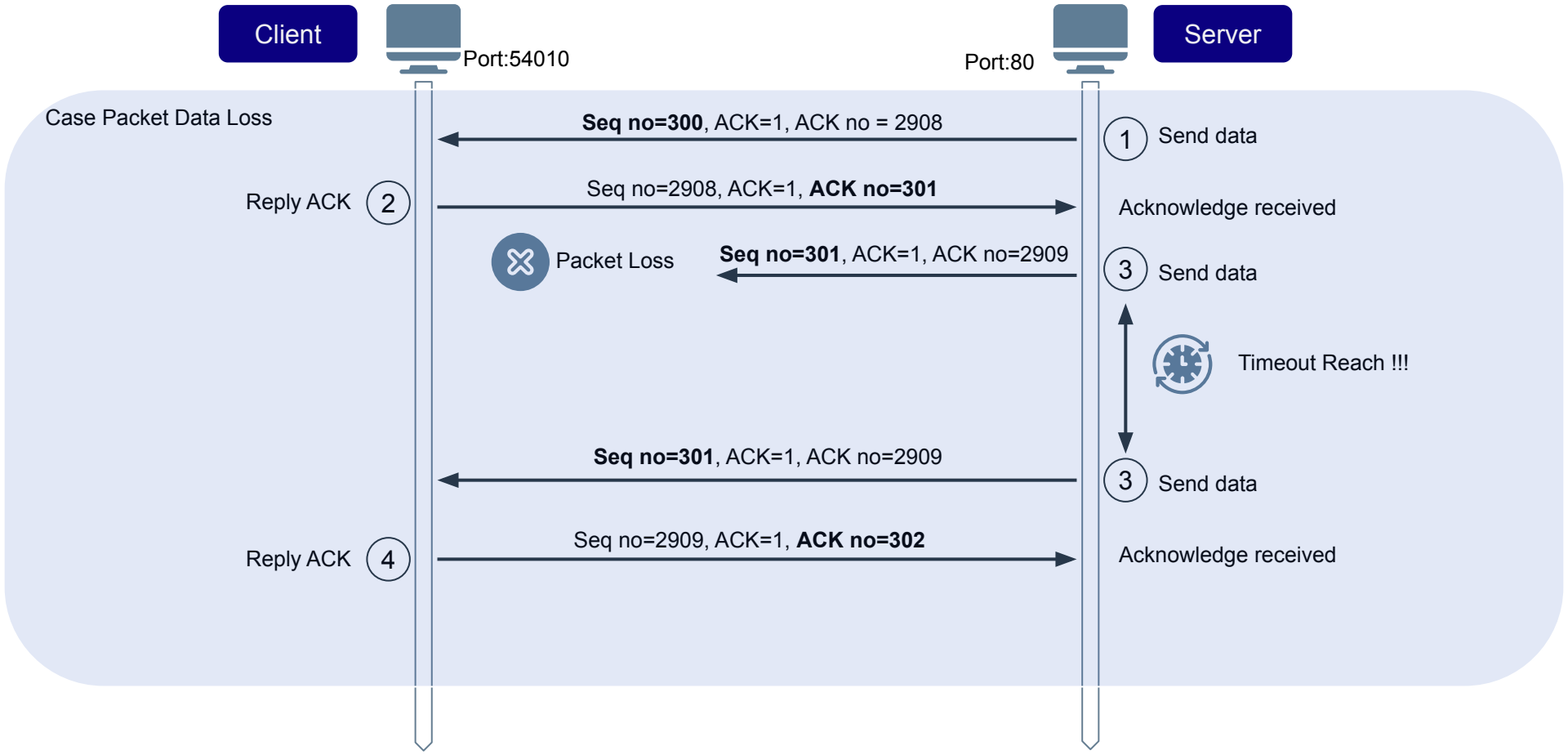
TCP Timeout and Retransmission

- So how can sender know and retransmission packet when it fail ?
- In fact, That no chance for sender to know about that failure. But on tcp connection, when sender send the packet to receiver. They will send timer of waiting reply from receiver.
- In generic case, receiver will reply packet (ACK) back in time. If no reply until timer reach “timeout”. Sender will assume that it need to retransmission
- Timeout on each os/application may be difference
  - Windows OS: ~120 - 3600 seconds
  - Linux OS: ~100 - 7200 seconds
  - Firewall: ~600 - 1800 seconds
  - Load Balance: ~60 - 300 seconds
- Timeout need to consideration and tuning for reduce retransmission rate that bring bad network quality

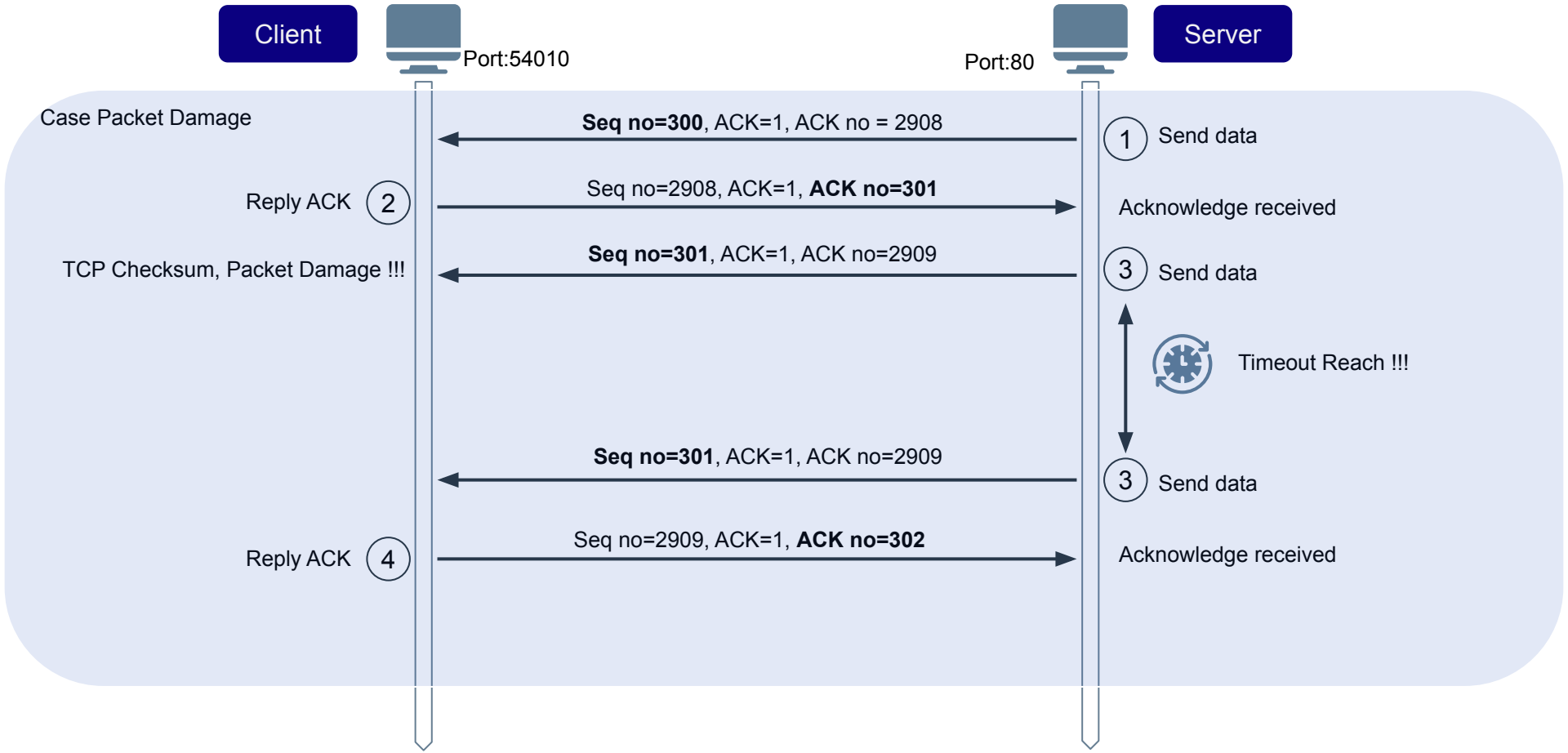
TCP Timeout and Retransmission



TCP Timeout and Retransmission



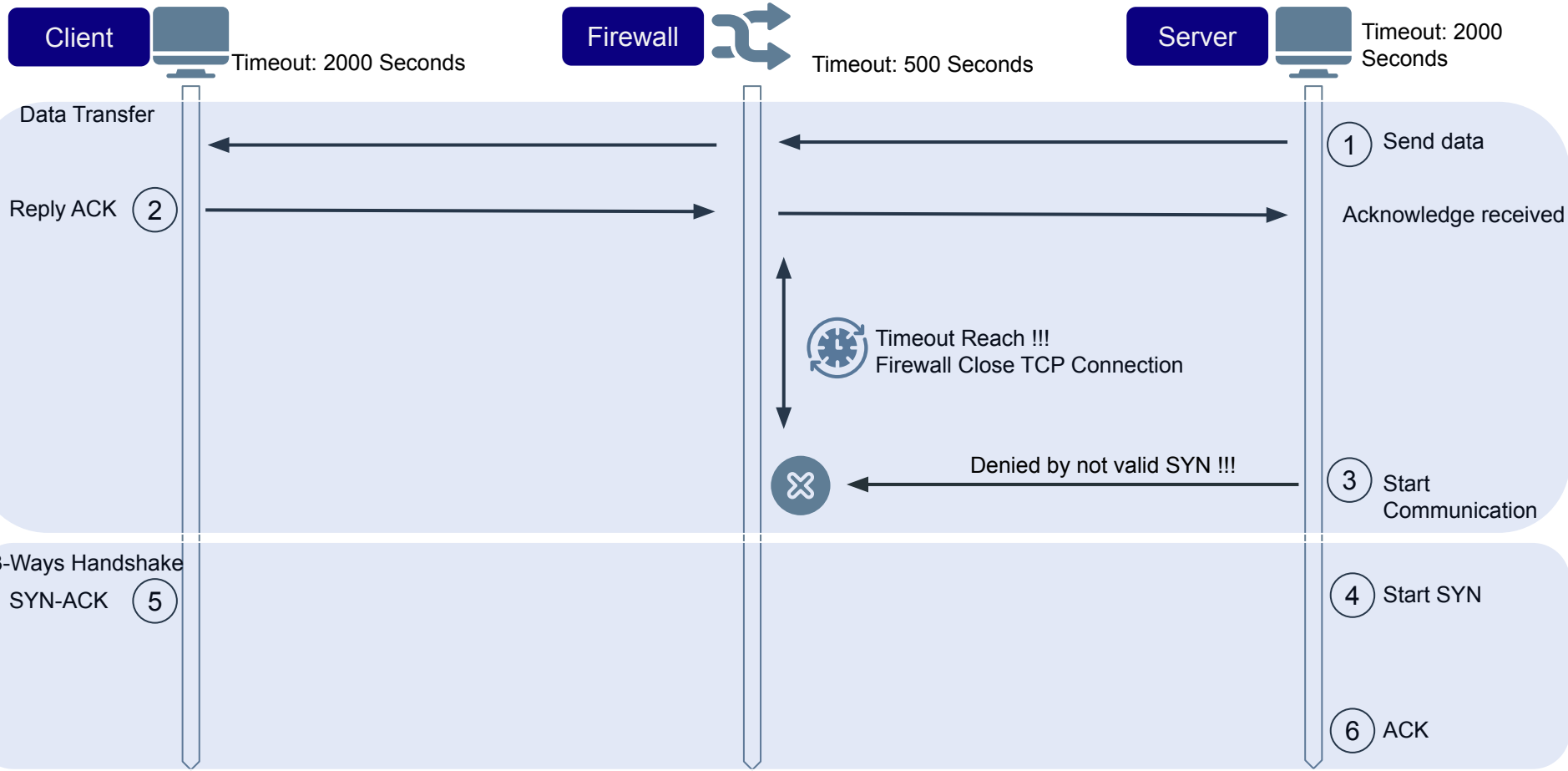
TCP Timeout and Retransmission



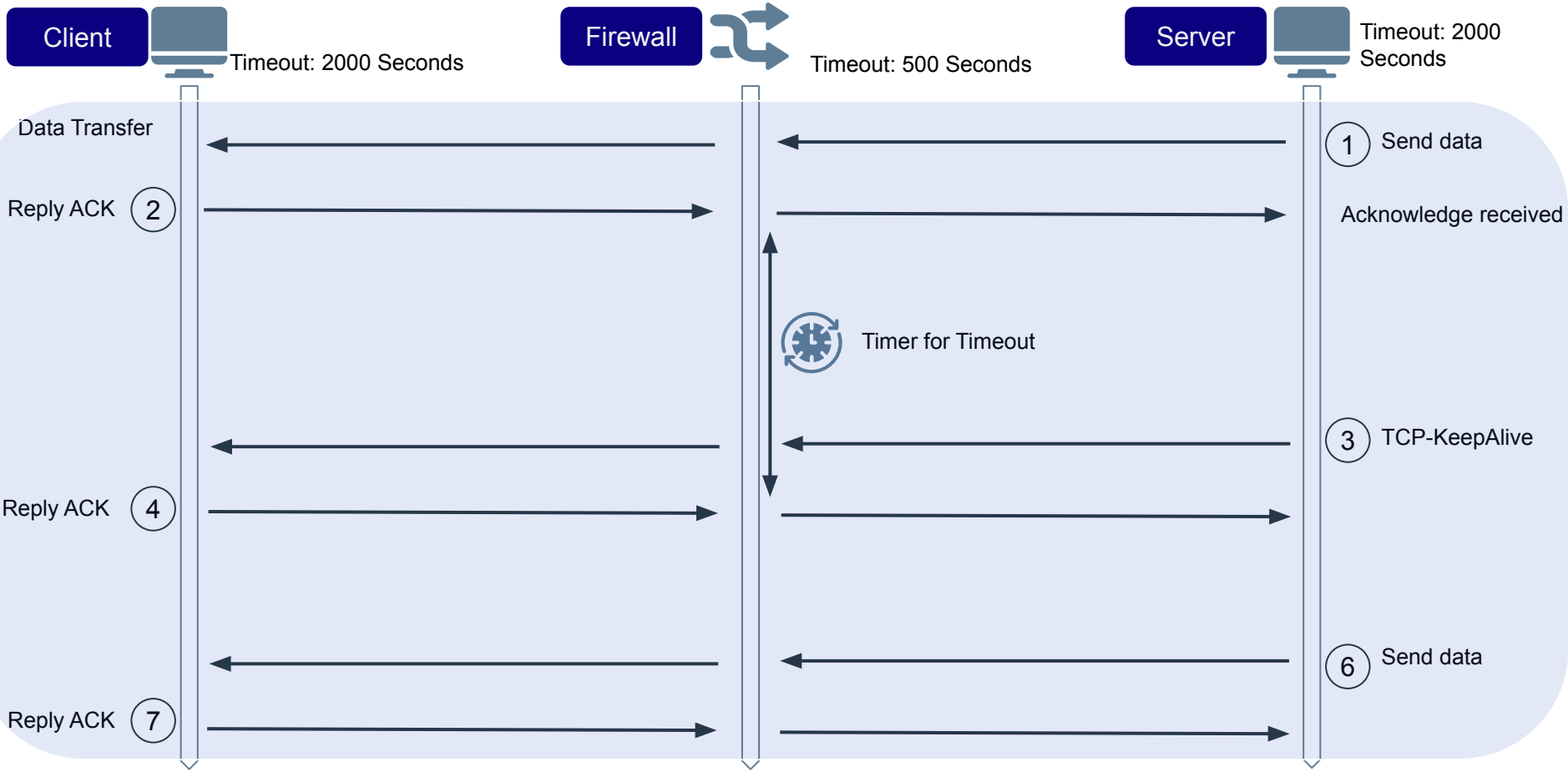
TCP Keep Alive

- On real world, Communication between sender and receiver will pass many intermediate device (L3 - L7)
- Each device will have their timeout identity (~30 - 6000 seconds)
- Many case tuning is impossible along end-to-end connection
- This will be classic case. When intermediate device was close tcp connection. That make sender and receiver waste time for re-establish connection again.
- Many retransmission - Many delay of application
- Many re-establish - Many bad on network traffic
- TCP have method to send packet “keep alive” for resolve this problem.
- After tcp handshake done. If idle connection is occuring. Tcp can configure for send

TCP Keep Alive



TCP Keep Alive



UDP Flow Opeate

- TCP keep all packet like child and make sure all packet can transmission to destination with complete and get confirm
- UDP is opposite of that. Why we need to confirm ?
- UDP is stateless communication. That mean they don't confirm completeness and let's upper layer check and verify this by themself
- Benefit of UDP is it reduce unnecessary packet to confirm and waiting
- Better network performance overall by this
- Many modern protocol on L7 have switch to use UDP instead TCP by reason above (Ex: HTTP/3 (QUIC))

UDP Flow Operate

- UDP packet (IPv4)

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source IPv4 Address																															
4	32	Destination IPv4 Address																															
8	64	Zeroes								Protocol								UDP Length															
12	96	Source Port																Destination Port															
16	128	Length																Checksum															
20	160+	Data																															

UDP Flow Operate

- UDP packet (IPv6)

Offsets Octet	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source IPv6 Address																															
4	32																																
8	64																																
12	96																																
16	128	Destination IPv6 Address																															
20	160																																
24	192																																
28	224																																
32	256	UDP Length																															
36	288	Zeroes																								Next Header = Protocol[10]							
40	320	Source Port																Destination Port															
44	352	Length																Checksum															
48	384+	Data																															

Workshop 1.5 UDP Flow Control

```
[ubuntu@ip-10-21-2-237:~$ sudo tcpdump -X -enni eth0 host 1.1.1.1 and udp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:49:43.778577 06:11:cc:15:31:04 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 97: 10.21.2.237.58260 > 1.1.1.1.53: 9324+ [1au] A? www.google.com. (55)
        0x0000:  4500 0053 c95c 0000 4011 a23a 0a15 02ed  E..S.\..@..:....
        0x0010:  0101 0101 e394 0035 003f 0f54 246c 0120  .......5.?.T$l..
        0x0020:  0001 0000 0000 0001 0377 7777 0667 6f6f  .........www.goo
        0x0030:  676c 6503 636f 6d00 0001 0001 0000 2910  gle.com.......).
        0x0040:  0000 0000 0000 0c00 0a00 0840 e560 9fd0  ...........@.`..
        0x0050:  371a 20                                  7..
17:49:43.781119 06:4d:1e:bf:fe:8e > 06:11:cc:15:31:04, ethertype IPv4 (0x0800), length 181: 1.1.1.1.53 > 10.21.2.237.58260: 9324 6/0/1 A 172.217.194.104, A 172.217.194.99, A 172.217.194.103, A 172.217.194.147, A 172.217.194.105, A 172.2
17.194.106 (139)
        0x0000:  4500 00a7 0db8 4000 3311 2a8b 0101 0101  E.....@.3.*.....
        0x0010:  0a15 02ed 0035 e394 0093 bf5b 246c 8180  .....5.....[$l..
        0x0020:  0001 0006 0000 0001 0377 7777 0667 6f6f  .........www.goo
        0x0030:  676c 6503 636f 6d00 0001 0001 c00c 0001  gle.com.........
        0x0040:  0001 0000 00a4 0004 acd9 c268 c00c 0001  ...........h....
        0x0050:  0001 0000 00a4 0004 acd9 c263 c00c 0001  ...........c....
        0x0060:  0001 0000 00a4 0004 acd9 c267 c00c 0001  ...........g....
        0x0070:  0001 0000 00a4 0004 acd9 c293 c00c 0001  ................
        0x0080:  0001 0000 00a4 0004 acd9 c269 c00c 0001  ...........i....
        0x0090:  0001 0000 00a4 0004 acd9 c26a 0000 2904  ...........j..).
        0x00a0:  d000 0000 0000 00                        .......
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel
ubuntu@ip-10-21-2-237:~$
```

```
[ubuntu@ip-10-21-2-237:~$ dig @1.1.1.1 www.google.com

; <<>> DiG 9.11.3-1ubuntu1.13-Ubuntu <<>> @1.1.1.1 www.google.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9324
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;www.google.com.                        IN      A

;; ANSWER SECTION:
www.google.com.         164     IN      A       172.217.194.104
www.google.com.         164     IN      A       172.217.194.99
www.google.com.         164     IN      A       172.217.194.103
www.google.com.         164     IN      A       172.217.194.147
www.google.com.         164     IN      A       172.217.194.105
www.google.com.         164     IN      A       172.217.194.106

;; Query time: 2 msec
;; SERVER: 1.1.1.1#53(1.1.1.1)
;; WHEN: Sat Dec 26 17:49:43 UTC 2020
;; MSG SIZE  rcvd: 139

ubuntu@ip-10-21-2-237:~$
```

TCPDump CheatSheet

Packet Capturing Options

Switch	Syntax	Description
-i any	tcpdump -i any	Capture from all interfaces
-i eth0	tcpdump -i eth0	Capture from specific interface (Ex Eth0)
-c	tcpdump -i eth0 -c 10	Capture first 10 packets and exit
-D	tcpdump -D	Show available interfaces
-A	tcpdump -i eth0 -A	Print in ASCII
-w	tcpdump -i eth0 -w tcpdump.txt	To save capture to a file
-r	tcpdump -r tcpdump.txt	Read and analyze saved capture file
-n	tcpdump -n -I eth0	Do not resolve host names
-nn	tcpdump -n -i eth0	Stop Domain name translation and lookups (Host names or port names)
tcp	tcpdump -i eth0 -c 10 -w tcpdump.pcap tcp	Capture TCP packets only
port	tcpdump -i eth0 port 80	Capture traffic from a defined port only
host	tcpdump host 192.168.1.100	Capture packets from specific host
net	tcpdump net 10.1.1.0/16	Capture files from network subnet
src	tcpdump src 10.1.1.100	Capture from a specific source address
dst	tcpdump dst 10.1.1.100	Capture from a specific destination address
<service>	tcpdump http	Filter traffic based on a port number for a service
<port>	tcpdump port 80	Filter traffic based on a service
port range	tcpdump portrange 21-125	Filter based on port range
-S	tcpdump -S http	Display entire packet
ipv6	tcpdunp -IPV6	Show only IPV6 packets
-d	tcpdump -d tcpdump.pcap	display human readable form in standard output
-F	tcpdump -F tcpdump.pcap	Use the given file as input for filter
-I	tcpdump -I eth0	set interface as monitor mode
-L	tcpdump -L	Display data link types for the interface
-N	tcpdump -N tcpdump.pcap	not printing domian names
-K	tcpdump -K tcpdump.pcap	Do not verify checksum
-p	tcpdump -p -i eth0	Not capturing in promiscuous mode

Logical Operators

Operator	Syntax	Example	Description
AND	and, &&	tcpdump -n src 192.168.1.1 and dst port 21	Combine filtering options
OR	or, ||	tcpdump dst 10.1.1.1 && !icmp	Either of the condition can match
EXCEPT	not, !	tcpdump dst 10.1.1.1 and not icmp	Negation of the condition
LESS	<	tcpdump <32	Shows packets size less than 32
GREATER	>	tcpdump >=32	Shows packets size greater than 32

Installation Commands

CENT OS and REDHAT	$ sudo yum install tcpdump
Fedora	$ dnf install tcpdump
Ubuntu, Debian and Linux Mint	#apt-get install tcpdump

Display / Output Options

Switch	Description
-q	Quite and less verbose mode display less details
-t	Do not print time stamp details in dump
-v	Little verbose output
-vv	More verbose output
-vvv	Most verbose output
-x	Print data and headers in HEX format
-xx	Print data with link headers in HEX format
-X	Print output in HEX and ASCII format **excluding** link headers
-XX	Print output in HEX and ASCII format **including** link headers
-e	Print Link (Ethernet) headers
-S	Print sequence numbers in exact format

Protocols

Ether, fddi, icmp ,ip, ip6 , ppp, radio, rarp, slip, tcp , udp, wlan

Common Commands with Protocols for Filtering Captures

src/ dst host (host name or IP)	Filter by source or destination IP address or host
ether src/ dst host (ethernet host name or IP)	Ethernet host filtering by source or destination
src/ dst net (subnet mask in CIDR)	Filter by subnet
tcp/udp src/dst port (port number)	Filter TCP or UDP packets by source or destination port
tcp/udp src/dst port range (port number range)	Filter TCP or UDP packets by source or destination port range
ether/ip broadcast	Filter for Ethernet or IP broadcasts
ether/ip multicast	Filter for Ethernet or IP multicasts

Ref: https://cdn.comparitech.com/wp-content/uploads/2019/06/tcpdump-cheat-sheet-1.pdf

Network Fundamental

DevOps Academy Project

DAY 2

Agenda

- **Day2**
- TCP/UDP flow control (OSI: L4, TCP/IP: L3)
  - Connection oriented vs Connectionless
  - TCP flow operate
    - TCP flow control and congestion control
      - TCP three way handshake and TCP terminate
      - Sliding windows
      - Timeout and Retransmission
      - TCP keep alive
  - UDP flow operate

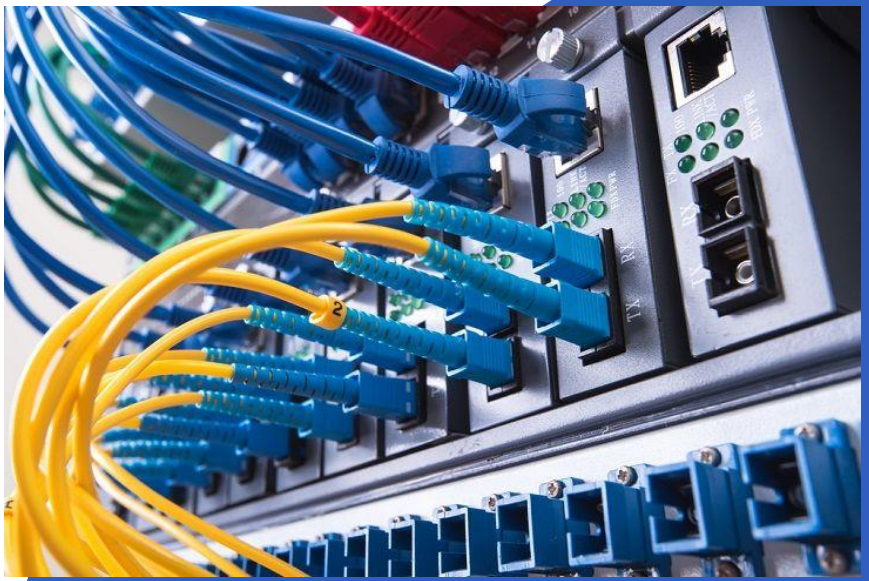
Agenda

- **Day2 (Continue)**
- Application layer flow control (OSI:L5 - L7, TCP/IP: L4)
  - DNS protocol (OSI: L6, TCP/IP: L4)
  - HTTP protocol (OSI: L7, TCP/IP: L4)
    - HTTP standard response code
    - HTTP generation
      - HTTP/0.9 - 1.1 (RFC:2616)
      - SPDY 2009, HTTP/2 (RFC:7540)
      - HTTP/3, QUIC (RFC:On review)
  - SSL/TLS protocol (OSI: L6+, TCP/IP: L4+)
      - SSL1.0, 2.0, 3.0 (RFC:6176, RFC:7568)
      - TLS1.0 (RFC: 2246)
      - TLS1.1 (RFC: 4346)
      - TLS1.2 (RFC: 5246)
      - TLS1.3 (RFC: 8446)

Agenda

- **Day2 (Continue)**
- Application layer flow control (OSI:L5 - L7, TCP/IP: L4)
  - Websocket and HTTP
- Advance network operation
  - ICMP operation
  - Linux IPTables operation
  - Linux IPVS load balance

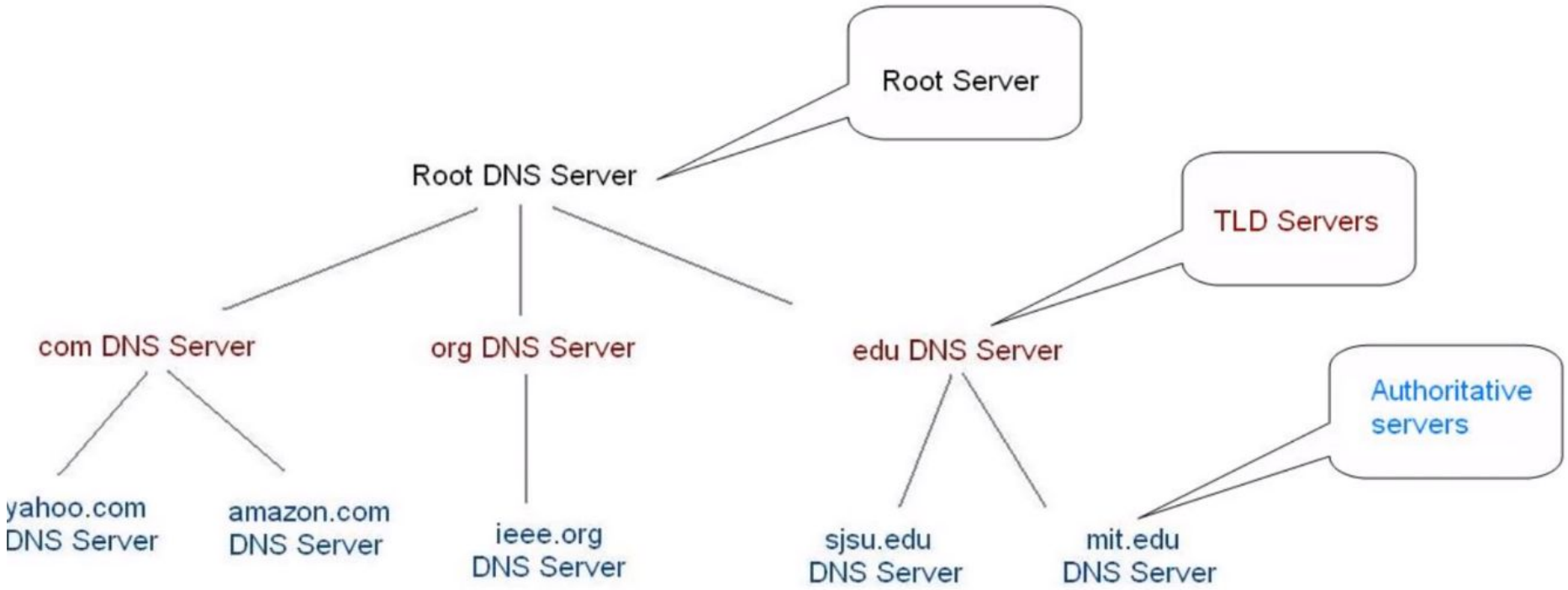
Application Layer Flow Control



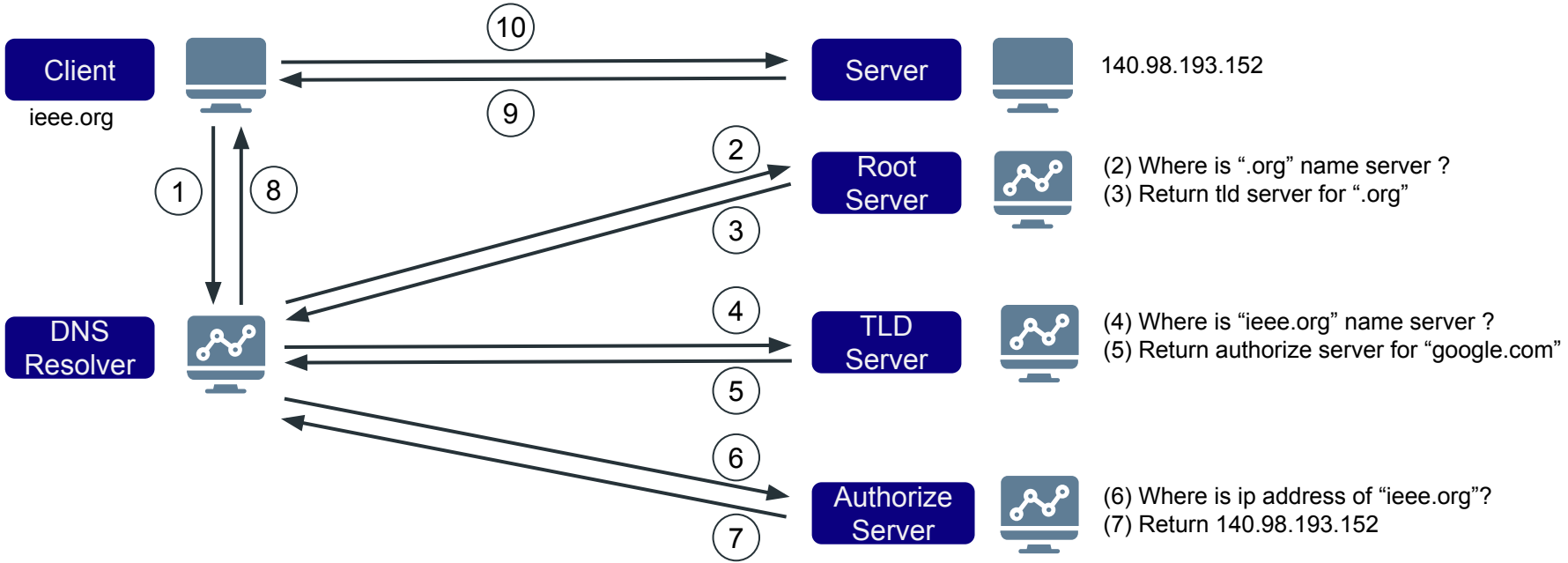
DNS Service(Domain Name Resolution)

- Hostname translator to ip address
- DNS process is start before computer can address real destination ip address
- Almost of service we use today will operate via DNS
- As all host need dns for operate. So dns task is quite heavy load on internet
- For rebalance this traffic dns service was designed to resolve as hierarchical
    - **DNS resolver/DNS recursor**: Non authoritative dns for accept dns query from client and give result back
    - **Root server**: Indexing, point TLD server for each domain
    - **TLD (top-level domain) server**: Contain root-hint dns
    - **Authoritative server**: Authoritative server who response back with ip address of specific name

DNS Service(Domain Name Resolution)



DNS Service(Domain Name Resolution)



Ref: https://www.cloudflare.com/th-th/learning/dns/what-is-dns/

DNS Service(Domain Name Resolution)

- DNS distributed db storing resource records (RR)
- Consist: **name**, **type**, class, **ttl**, rdlength, rdata
- TTL (time to live, seconds): remain how long of record need to update
- Type of record
  - **A,AAAA (address record)**: name of host, value = ip address
  - **CNAME (canonical)**: alias of host, value = name
  - **MX (mail exchange)**: mail server, value = name
  - **SOA (start of authority)**: authorize server of record, value=name
  - **NS (name server)**: name of delegate server (Ex: sdn.google.com), value = ip address
  - **TXT**: human readable for multi purpose, value = text
  - etc.

Ref: https://en.wikipedia.org/wiki/List_of_DNS_record_types

DNS Service(Domain Name Resolution)

Field	Description	Length (octets)
NAME	Name of the node to which this record pertains	Variable
TYPE	Type of RR in numeric form (e.g., 15 for MX RRs)	2
CLASS	Class code	2
TTL	Count of seconds that the RR stays valid (The maximum is $2^{31}-1$, which is about 68 years)	4
RDLENGTH	Length of RDATA field (specified in octets)	2
RDATA	Additional RR-specific data	Variable, as per RDLENGTH

Ref: https://en.wikipedia.org/wiki/List_of_DNS_record_types

DNS Service(Domain Name Resolution)

```
praparns-MacBook-Pro:~ praparn$ nslookup -debug -type=a google.com 8.8.8.8
Server:         8.8.8.8
Address:        8.8.8.8#53

------------
    QUESTIONS:
        google.com, type = A, class = IN
    ANSWERS:
    ->  google.com
        internet address = 216.58.196.46
        ttl = 252
    AUTHORITY RECORDS:
    ADDITIONAL RECORDS:
------------
Non-authoritative answer:
Name:   google.com
Address: 216.58.196.46

praparns-MacBook-Pro:~ praparn$ nslookup -debug -type=aaaa google.com 8.8.8.8
Server:         8.8.8.8
Address:        8.8.8.8#53

------------
    QUESTIONS:
        google.com, type = AAAA, class = IN
    ANSWERS:
    ->  google.com
        has AAAA address 2404:6800:4001:80c::200e
        ttl = 259
    AUTHORITY RECORDS:
    ADDITIONAL RECORDS:
------------
Non-authoritative answer:
google.com      has AAAA address 2404:6800:4001:80c::200e

Authoritative answers can be found from:

praparns-MacBook-Pro:~ praparn$
```

```
praparns-MacBook-Pro:~ praparn$ nslookup -debug -type=soa google.com 8.8.8.8
Server:         8.8.8.8
Address:        8.8.8.8#53

------------
    QUESTIONS:
        google.com, type = SOA, class = IN
    ANSWERS:
    ->  google.com
        origin = ns1.google.com
        mail addr = dns-admin.google.com
        serial = 346996212
        refresh = 900
        retry = 900
        expire = 1800
        minimum = 60
        ttl = 2
    AUTHORITY RECORDS:
    ADDITIONAL RECORDS:
------------
Non-authoritative answer:
google.com
        origin = ns1.google.com
        mail addr = dns-admin.google.com
        serial = 346996212
        refresh = 900
        retry = 900
        expire = 1800
        minimum = 60

Authoritative answers can be found from:
```

DNS Service(Domain Name Resolution)

```
praparns-MacBook-Pro:~ praparn$ nslookup -debug -type=mx google.com 8.8.8.8
Server:         8.8.8.8
Address:        8.8.8.8#53

------------
    QUESTIONS:
        google.com, type = MX, class = IN
    ANSWERS:
    ->  google.com
        mail exchanger = 50 alt4.aspmx.l.google.com.
        ttl = 53
    ->  google.com
        mail exchanger = 30 alt2.aspmx.l.google.com.
        ttl = 53
    ->  google.com
        mail exchanger = 40 alt3.aspmx.l.google.com.
        ttl = 53
    ->  google.com
        mail exchanger = 10 aspmx.l.google.com.
        ttl = 53
    ->  google.com
        mail exchanger = 20 alt1.aspmx.l.google.com.
        ttl = 53
    AUTHORITY RECORDS:
    ADDITIONAL RECORDS:
------------
Non-authoritative answer:
google.com      mail exchanger = 50 alt4.aspmx.l.google.com.
google.com      mail exchanger = 30 alt2.aspmx.l.google.com.
google.com      mail exchanger = 40 alt3.aspmx.l.google.com.
google.com      mail exchanger = 10 aspmx.l.google.com.
google.com      mail exchanger = 20 alt1.aspmx.l.google.com.

Authoritative answers can be found from:
```

```
praparns-MacBook-Pro:~ praparn$ nslookup -debug -type=txt google.com 8.8.8.8
Server:         8.8.8.8
Address:        8.8.8.8#53

------------
    QUESTIONS:
        google.com, type = TXT, class = IN
    ANSWERS:
    ->  google.com
        text = "globalsign-smime-dv=CDYX+XFHUw2wml6/Gb8+59BsH31KzUr6c1l2BPvqKX8="
        ttl = 3599
    ->  google.com
        text = "docusign=1b0a6754-49b1-4db5-8540-d2c12664b289"
        ttl = 299
    ->  google.com
        text = "docusign=05958488-4752-4ef2-95eb-aa7ba8a3bd0e"
        ttl = 299
    ->  google.com
        text = "facebook-domain-verification=22rm551cu4k0ab0bxsw536tlds4h95"
        ttl = 3599
    ->  google.com
        text = "v=spf1 include:_spf.google.com ~all"
        ttl = 3599
    AUTHORITY RECORDS:
    ADDITIONAL RECORDS:
------------
Non-authoritative answer:
google.com      text = "globalsign-smime-dv=CDYX+XFHUw2wml6/Gb8+59BsH31KzUr6c1l2BPvqKX8="
google.com      text = "docusign=1b0a6754-49b1-4db5-8540-d2c12664b289"
google.com      text = "docusign=05958488-4752-4ef2-95eb-aa7ba8a3bd0e"
google.com      text = "facebook-domain-verification=22rm551cu4k0ab0bxsw536tlds4h95"
google.com      text = "v=spf1 include:_spf.google.com ~all"

Authoritative answers can be found from:

praparns-MacBook-Pro:~ praparn$
```

DNS Service(Domain Name Resolution)

```
praparns-MacBook-Pro:~ praparn$ nslookup -debug -type=any google.com 8.8.8.8
;; Truncated, retrying in TCP mode.
Server:         8.8.8.8
Address:        8.8.8.8#53

------------
    QUESTIONS:
        google.com, type = ANY, class = IN
    ANSWERS:
    ->  google.com
        internet address = 74.125.24.139
        ttl = 265
    ->  google.com
        internet address = 74.125.24.102
        ttl = 265
    ->  google.com
        internet address = 74.125.24.138
        ttl = 265
    ->  google.com
        internet address = 74.125.24.101
        ttl = 265
    ->  google.com
        internet address = 74.125.24.113
        ttl = 265
    ->  google.com
        internet address = 74.125.24.100
        ttl = 265
    ->  google.com
        has AAAA address 2404:6800:4003:c03::65
        ttl = 265
    ->  google.com
        has AAAA address 2404:6800:4003:c03::71
        ttl = 265
    ->  google.com
        has AAAA address 2404:6800:4003:c03::66
        ttl = 265
    ->  google.com
        has AAAA address 2404:6800:4003:c03::8b
        ttl = 265
    ->  google.com
        mail exchanger = 20 alt1.aspmx.l.google.com.
        ttl = 565
    ->  google.com
        mail exchanger = 10 aspmx.l.google.com.
        ttl = 565
    ->  google.com
        nameserver = ns1.google.com.
        ttl = 21565
    ->  google.com
        mail exchanger = 40 alt3.aspmx.l.google.com.
        ttl = 565
    ->  google.com
        text = "facebook-domain-verification=22rm551cu4k0ab0bxsw536tlds4h95"
    ->  google.com
        nameserver = ns3.google.com.
        ttl = 21565
    ->  google.com
        text = "globalsign-smime-dv=CDYX+XFHUw2wml6/Gb8+59BsH31KzUr6c1l2BPvqKX8="
        ttl = 3565
    ->  google.com
        text = "docusign=05958488-4752-4ef2-95eb-aa7ba8a3bd0e"
        ttl = 265
    ->  google.com
        nameserver = ns2.google.com.
        ttl = 21565
    ->  google.com
        text = "v=spf1 include:_spf.google.com ~all"
        ttl = 3565
    ->  google.com
        nameserver = ns4.google.com.
        ttl = 21565
    ->  google.com
        text = "docusign=1b0a6754-49b1-4db5-8540-d2c12664b289"
        ttl = 265
    ->  google.com
        mail exchanger = 30 alt2.aspmx.l.google.com.
        ttl = 565
    ->  google.com
        rdata_257 = 0 issue "pki.goog"
        ttl = 21565
    ->  google.com
        mail exchanger = 50 alt4.aspmx.l.google.com.
        ttl = 565
    ->  google.com
        origin = ns1.google.com
        mail addr = dns-admin.google.com
        serial = 346996212
        refresh = 900
        retry = 900
        expire = 1800
        minimum = 60
        ttl = 25
    AUTHORITY RECORDS:
    ADDITIONAL RECORDS:
------------
Non-authoritative answer:
Name:   google.com
Address: 74.125.24.139
Name:   google.com
Address: 74.125.24.102
Name:   google.com
Address: 74.125.24.138
Name:   google.com
Address: 74.125.24.101
Name:   google.com
Address: 74.125.24.113
```

DNS Service(Domain Name Resolution)

```
praparns-MacBook-Pro:~ praparn$ dig +nocmd +noall +answer A google.com @8.8.8.8
google.com.             146     IN      A       172.217.27.238
praparns-MacBook-Pro:~ praparn$ dig +nocmd +noall +answer A google.com @8.8.8.8
google.com.             235     IN      A       172.217.174.174
praparns-MacBook-Pro:~ praparn$ dig +nocmd +noall +answer A google.com @8.8.8.8
google.com.             239     IN      A       172.217.174.174
praparns-MacBook-Pro:~ praparn$ dig +nocmd +noall +answer A google.com @8.8.8.8
google.com.             129     IN      A       216.58.196.46
praparns-MacBook-Pro:~ praparn$ dig +nocmd +noall +answer A google.com @8.8.8.8
google.com.             176     IN      A       172.217.31.78
praparns-MacBook-Pro:~ praparn$ dig +nocmd +noall +answer A google.com @8.8.8.8
google.com.             16      IN      A       172.217.174.174
praparns-MacBook-Pro:~ praparn$
praparns-MacBook-Pro:~ praparn$ dig +nocmd +noall +answer AAAA google.com @8.8.8.8
google.com.             58      IN      AAAA    2404:6800:4001:805::200e
praparns-MacBook-Pro:~ praparn$ dig +nocmd +noall +answer AAAA google.com @8.8.8.8
google.com.             31      IN      AAAA    2404:6800:4001:80f::200e
praparns-MacBook-Pro:~ praparn$ dig +nocmd +noall +answer AAAA google.com @8.8.8.8
google.com.             116     IN      AAAA    2404:6800:4001:80e::200e
praparns-MacBook-Pro:~ praparn$ dig +nocmd +noall +answer AAAA google.com @8.8.8.8
google.com.             137     IN      AAAA    2404:6800:4001:80f::200e
praparns-MacBook-Pro:~ praparn$ dig +nocmd +noall +answer AAAA google.com @8.8.8.8
google.com.             108     IN      AAAA    2404:6800:4001:80e::200e
praparns-MacBook-Pro:~ praparn$
```

```
praparns-MacBook-Pro:~ praparn$ dig +nocmd +noall +answer SOA google.com @8.8.8.8
google.com.             43      IN      SOA     ns1.google.com. dns-admin.google.com. 346996212 900 900 1800 60
praparns-MacBook-Pro:~ praparn$ dig +nocmd +noall +answer NS google.com @8.8.8.8
google.com.             8677    IN      NS      ns1.google.com.
google.com.             8677    IN      NS      ns4.google.com.
google.com.             8677    IN      NS      ns2.google.com.
google.com.             8677    IN      NS      ns3.google.com.
praparns-MacBook-Pro:~ praparn$
```

```
praparns-MacBook-Pro:~ praparn$ dig +nocmd +noall +answer MX google.com @8.8.8.8
google.com.             23      IN      MX      20 alt1.aspmx.l.google.com.
google.com.             23      IN      MX      10 aspmx.l.google.com.
google.com.             23      IN      MX      30 alt2.aspmx.l.google.com.
google.com.             23      IN      MX      50 alt4.aspmx.l.google.com.
google.com.             23      IN      MX      40 alt3.aspmx.l.google.com.
praparns-MacBook-Pro:~ praparn$ dig +nocmd +noall +answer TXT google.com @8.8.8.8
google.com.             299     IN      TXT     "docusign=1b0a6754-49b1-4db5-8540-d2c12664b289"
google.com.             3599    IN      TXT     "facebook-domain-verification=22rm551cu4k0ab0bxsw536tlds4h95"
google.com.             3599    IN      TXT     "v=spf1 include:_spf.google.com ~all"
google.com.             299     IN      TXT     "docusign=05958488-4752-4ef2-95eb-aa7ba8a3bd0e"
google.com.             3599    IN      TXT     "globalsign-smime-dv=CDYX+XFHUw2wml6/Gb8+59BsH31KzUr6c1l2BPvqKX8="
praparns-MacBook-Pro:~ praparn$
```

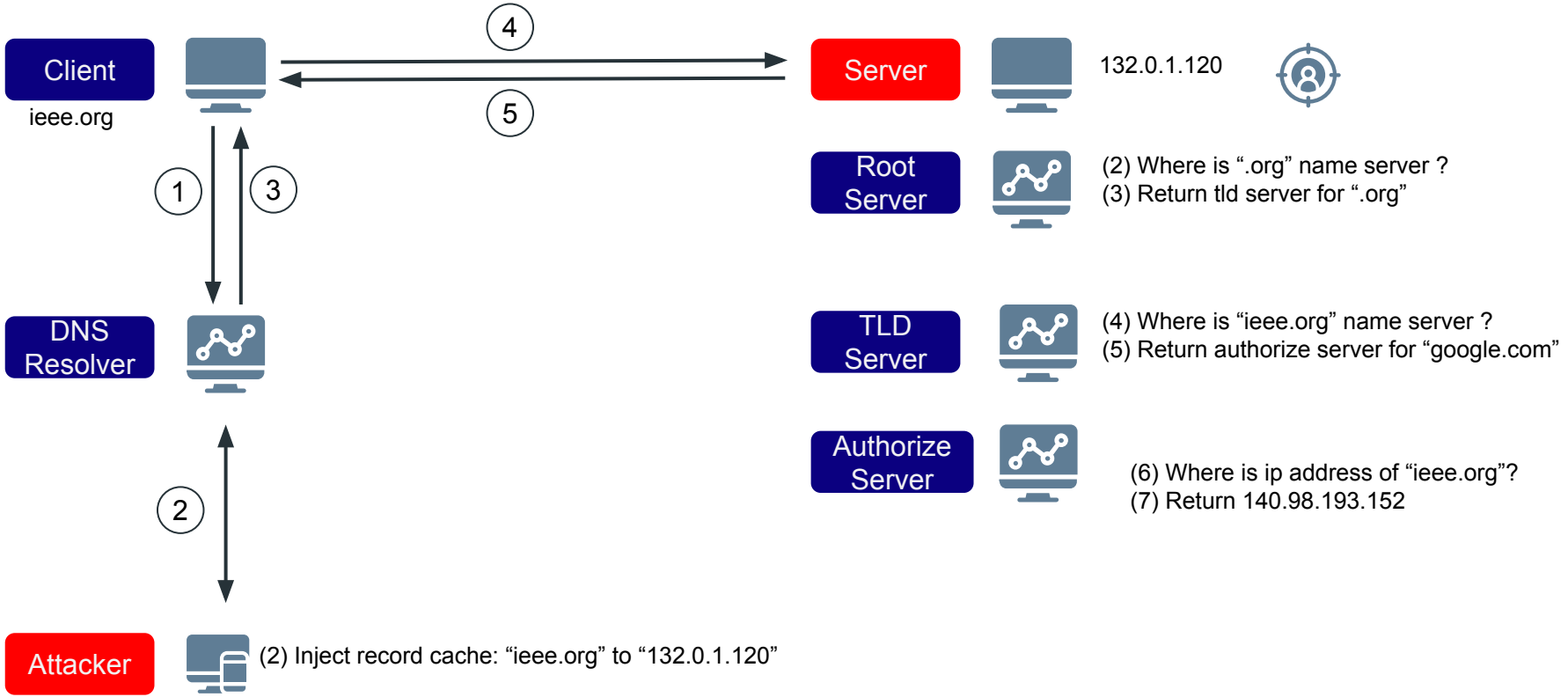
```
praparns-MacBook-Pro:~ praparn$ dig +nocmd +noall +answer ANY google.com @8.8.8.8
google.com.             279     IN      A       172.253.118.102
google.com.             279     IN      A       172.253.118.100
google.com.             279     IN      A       172.253.118.101
google.com.             279     IN      A       172.253.118.138
google.com.             279     IN      A       172.253.118.139
google.com.             279     IN      A       172.253.118.113
google.com.             279     IN      AAAA    2404:6800:4003:c05::8b
google.com.             279     IN      AAAA    2404:6800:4003:c05::66
google.com.             279     IN      AAAA    2404:6800:4003:c05::71
google.com.             279     IN      AAAA    2404:6800:4003:c05::64
google.com.             3579    IN      TXT     "facebook-domain-verification=22rm551cu4k0ab0bxsw536tlds4h95"
google.com.             579     IN      MX      30 alt2.aspmx.l.google.com.
google.com.             21579   IN      CAA     0 issue "pki.goog"
google.com.             39      IN      SOA     ns1.google.com. dns-admin.google.com. 346996212 900 900 1800 60
google.com.             21579   IN      NS      ns2.google.com.
google.com.             579     IN      MX      20 alt1.aspmx.l.google.com.
google.com.             3579    IN      TXT     "globalsign-smime-dv=CDYX+XFHUw2wml6/Gb8+59BsH31KzUr6c1l2BPvqKX8="
google.com.             21579   IN      NS      ns4.google.com.
google.com.             579     IN      MX      40 alt3.aspmx.l.google.com.
google.com.             21579   IN      NS      ns1.google.com.
google.com.             279     IN      TXT     "docusign=05958488-4752-4ef2-95eb-aa7ba8a3bd0e"
google.com.             279     IN      TXT     "docusign=1b0a6754-49b1-4db5-8540-d2c12664b289"
google.com.             21579   IN      NS      ns3.google.com.
google.com.             579     IN      MX      50 alt4.aspmx.l.google.com.
google.com.             579     IN      MX      10 aspmx.l.google.com.
google.com.             3579    IN      TXT     "v=spf1 include:_spf.google.com ~all"
praparns-MacBook-Pro:~ praparn$
```

DNS Service(Domain Name Resolution)

- When dns success for query record. They will keep record on “cache” for use in next time and reduce re-work.
- TLD server typical cache authorize name server
- DNS resolver (local dns)/Host also cache dns record for improve performance.
- DNS cache will keep until reach ttl (time to live) timer before start to query dns record again.
- This cache mechanism will be the one of vulnerability call “DNS cache poisoning”
- Attacker will inject fake record to DNS resolver for redirect traffic from client to attacker website!!!

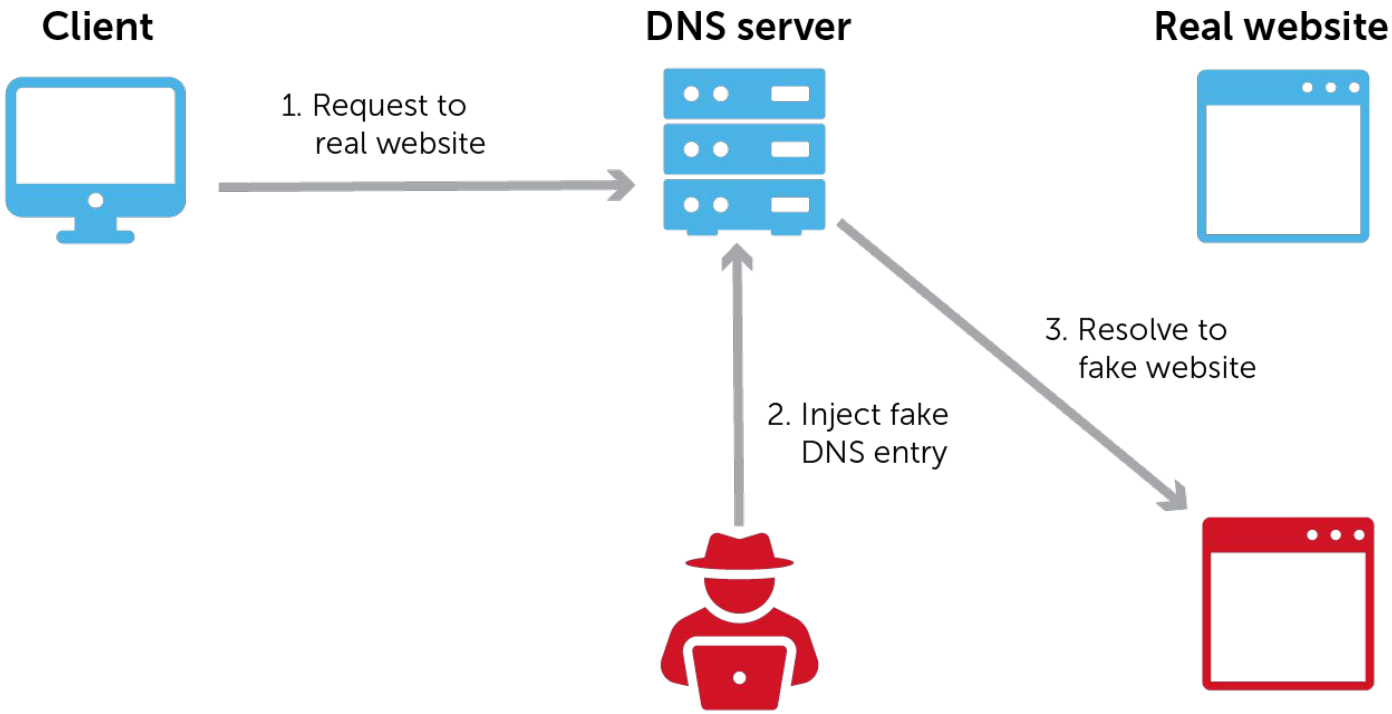
Ref: https://www.cloudflare.com/th-th/learning/dns/what-is-dns/

DNS Service(Domain Name Resolution)

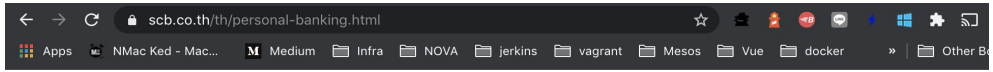


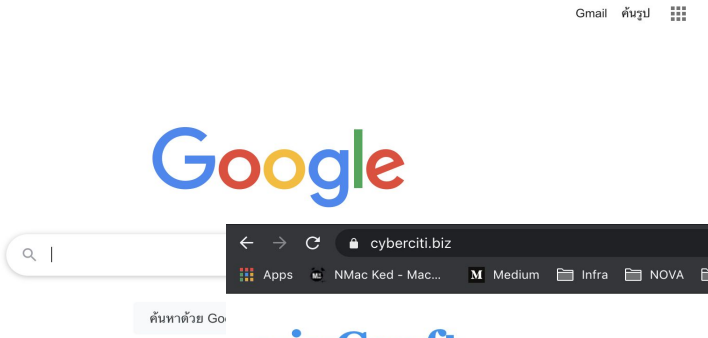
Ref: https://www.cloudflare.com/th-th/learning/dns/what-is-dns/

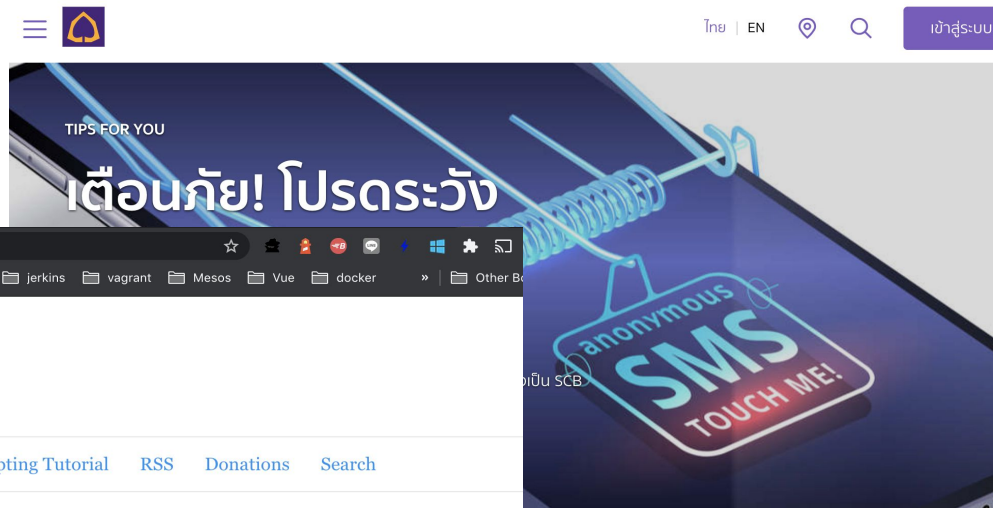
DNS Service(Domain Name Resolution)



Workshop 2.1 DNS Service







nixCraft

Linux Tips, Hacks, Tutorials, And Ideas In Blog

↔ Home Howtos & Tutorials Linux Shell Scripting Tutorial RSS Donations Search

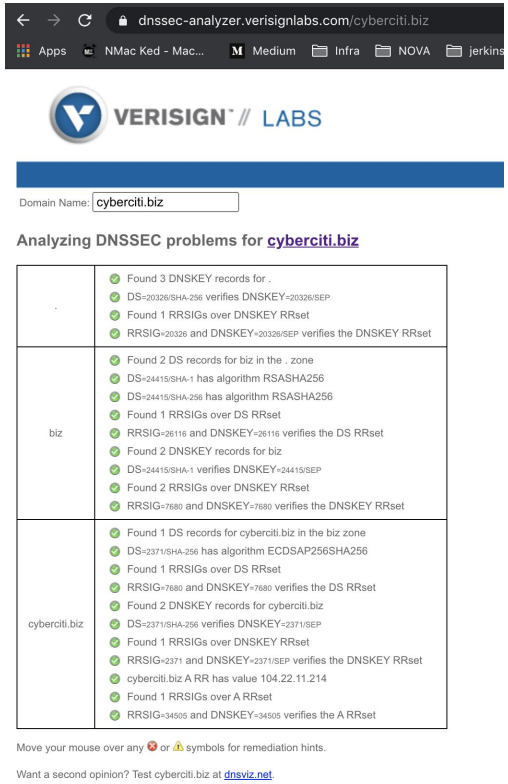
bpytop – Awesome Linux, macOS and FreeBSD resource monitor

The bashtop is an impressive Linux resource monitor that shows usage and stats for processor, memory, disks, and network. However, it suffers from bash itself, and cross-platform support is a nightmare. Now we have the Python port of bashtop. We can use a resource monitor that shows usage and stats for CPU, RAM, SSD (hard disk), network, and processes information in a lovely format.

[continue reading...]

Ad

Workshop 2.1 DNS Service




```
praparns-MacBook-Pro:.ssh praparn$ dig +dnssec cyberciti.biz

; <<>> DiG 9.10.6 <<>> +dnssec cyberciti.biz
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49030
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;cyberciti.biz.                 IN      A

;; ANSWER SECTION:
cyberciti.biz.          300     IN      A       104.22.10.214
cyberciti.biz.          300     IN      A       104.22.11.214
cyberciti.biz.          300     IN      A       172.67.7.239
cyberciti.biz.          300     IN      RRSIG   A 13 2 300 20201228073708 20201226053708 34505 cyberciti.biz. 2KVP3tYhYm6PSB1RTrr8mI3V1qv7a6CXN4BiGiCKxaUeDNGQIS/15R1A SniAag2jmIoiqI0Wp1poyFG4u4v+UA==

;; Query time: 52 msec
;; SERVER: 1.1.1.1#53(1.1.1.1)
;; WHEN: Sun Dec 27 13:37:07 +07 2020
;; MSG SIZE  rcvd: 199

praparns-MacBook-Pro:.ssh praparn$ dig +dnssec google.com

; <<>> DiG 9.10.6 <<>> +dnssec google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6459
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             40      IN      A       172.217.166.142

;; Query time: 30 msec
;; SERVER: 1.1.1.1#53(1.1.1.1)
;; WHEN: Sun Dec 27 13:37:16 +07 2020
;; MSG SIZE  rcvd: 55

praparns-MacBook-Pro:.ssh praparn$
```

DIG option

```
praparns-MacBook-Pro:.ssh praparn$ dig -help
Usage:  dig [@global-server] [domain] [q-type] [q-class] {q-opt}
            {global-d-opt} host [@local-server] {local-d-opt}
            [ host [@local-server] {local-d-opt} [...]]
Where:  domain    is in the Domain Name System
        q-class  is one of (in,hs,ch,...) [default: in]
        q-type   is one of (a,any,mx,ns,soa,hinfo,axfr,txt,...) [default:a]
                 (Use ixfr=version for type ixfr)
        q-opt    is one of:
                 -4                  (use IPv4 query transport only)
                 -6                  (use IPv6 query transport only)
                 -b address[#port]   (bind to source address/port)
                 -c class            (specify query class)
                 -f filename         (batch mode)
                 -i                  (use IP6.INT for IPv6 reverse lookups)
                 -k keyfile          (specify tsig key file)
                 -m                  (enable memory usage debugging)
                 -p port             (specify port number)
                 -q name             (specify query name)
                 -t type             (specify query type)
                 -u                  (display times in usec instead of msec)
                 -x dot-notation     (shortcut for reverse lookups)
                 -y [hmac:]name:key  (specify named base64 tsig key)
        d-opt    is of the form +keyword[=value], where keyword is:
                 +[no]aaonly         (Set AA flag in query (+[no]aaflag))
                 +[no]additional     (Control display of additional section)
                 +[no]adflag         (Set AD flag in query (default on))
                 +[no]all            (Set or clear all display flags)
                 +[no]answer         (Control display of answer section)
                 +[no]authority      (Control display of authority section)
                 +[no]besteffort     (Try to parse even illegal messages)
                 +bufsize=###        (Set EDNS0 Max UDP packet size)
                 +[no]cdflag         (Set checking disabled flag in query)
                 +[no]cl             (Control display of class in records)
                 +[no]cmd            (Control display of command line)
                 +[no]comments       (Control display of comment lines)
                 +[no]crypto         (Control display of cryptographic fields in records)
                 +[no]defname        (Use search list (+[no]search))
                 +[no]dnssec         (Request DNSSEC records)
                 +domain=###         (Set default domainname)
                 +[no]edns[=###]     (Set EDNS version) [0]
                 +ednsflags=###      (Set EDNS flag bits)
                 +[no]ednsnegotiation (Set EDNS version negotiation)
                 +ednsopt=###[:value] (Send specified EDNS option)
                 +noednsopt          (Clear list of +ednsopt options)
                 +[no]expire         (Request time to expire)
                 +[no]fail           (Don't try next server on SERVFAIL)
                 +[no]identify       (ID responders in short answers)
                 +[no]idnout         (convert IDN response)
                 +[no]ignore         (Don't revert to TCP for TC responses.)
                 +[no]keepopen       (Keep the TCP socket open between queries)
                 +[no]multiline      (Print records in an expanded format)
                 +ndots=###          (Set search NDOTS value)
                 +[no]nsid           (Request Name Server ID)
                 +[no]nssearch       (Search all authoritative nameservers)
                 +[no]onesoa         (AXFR prints only one soa record)
                 +[no]opcode=###     (Set the opcode of the request)
                 +[no]qr             (Print question before sending)
                 +[no]question       (Control display of question section)
                 +[no]recurse        (Recursive mode)
                 +retry=###          (Set number of UDP retries) [2]
                 +[no]rrcomments     (Control display of per-record comments)
                 +[no]search         (Set whether to use searchlist)
                 +[no]short          (Display nothing except short
                                      form of answer)
                 +[no]showsearch     (Search with intermediate results)
                 +[no]split=##       (Split hex/base64 fields into chunks)
                 +[no]stats          (Control display of statistics)
                 +subnet=addr        (Set edns-client-subnet option)
                 +[no]tcp            (TCP mode (+[no]vc))
                 +time=###           (Set query timeout) [5]
                 +[no]trace          (Trace delegation down from root [+dnssec])
                 +tries=###          (Set number of UDP attempts) [3]
                 +[no]ttlid          (Control display of ttls in records)
                 +[no]vc             (TCP mode (+[no]tcp))
        global d-opts and servers (before host name) affect all queries.
        local d-opts and servers (after host name) affect only that lookup.
        -h                           (print help and exit)
        -v                           (print version and exit)
praparns-MacBook-Pro:.ssh praparn$
```

HTTP Service(Hypertext Transfer Protocol)

- HTTP service was invented and announcement first standard on 1991. With only 1 method as “GET /index.html” and response with HTML (Hypertext Markup Language)
- Stateless connection for support scalability.
- Operate on Client-Server architecture with “request-response”. **Client need to start connection.**
- Design by text base transfer on first and improvement until now.
- Be standard protocol of web traffic on internet

HTTP Service(Hypertext Transfer Protocol)

- HTTP response code was issued by server to response client's request start on HTTP/1.1 (RFC 7231)
- All modernize application will use response code for indicate event trigger and response
- HTTP response code can categories in 5 group
  - 1xx: Information: Request was received, Continue process
  - 2xx: Success: Request was received, understood, accepted
  - 3xx: Redirect: Further action needs to be taken in order to complete the request
  - 4xx: Client Error: The request contains bad syntax or cannot be fulfilled
  - 5xx: Server Error: the server failed to fulfil an apparently valid request

HTTP Service(Hypertext Transfer Protocol)

-

1XX Informational	
100	Continue
101	Switching Protocols
102	Processing

2XX Success	
200	OK
201	Created
202	Accepted
203	Non-authoritative Information
204	No Content
205	Reset Content
206	Partial Content
207	Multi-Status
208	Already Reported
226	IM Used

3XX Redirectional	
300	Multiple Choices
301	Moved Permanently
302	Found
303	See Other
304	Not Modified
305	Use Proxy
307	Temporary Redirect
308	Permanent Redirect

4XX Client Error	
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
407	Proxy Authentication Required
408	Request Timeout

4XX Client Error Continued	
409	Conflict
410	Gone
411	Length Required
412	Precondition Failed
413	Payload Too Large
414	Request-URI Too Long
415	Unsupported Media Type
416	Requested Range Not Satisfiable
417	Expectation Failed
418	I'm a teapot
421	Misdirected Request
422	Unprocessable Entity
423	Locked
424	Failed Dependency
426	Upgrade Required
428	Precondition Required
429	Too Many Requests
431	Request Header Fields Too Large
444	Connection Closed Without Response
451	Unavailable For Legal Reasons
499	Client Closed Request

5XX Server Error	
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout
505	HTTP Version Not Supported
506	Variant Also Negotiates
507	Insufficient Storage
508	Loop Detected
510	Not Extended
511	Network Authentication Required
599	Network Connect Timeout Error

HTTP Service(Hypertext Transfer Protocol)

- **Generation of HTTP**
  - **1st generation (Text base)**
    - HTTP/0.9
    - HTTP/1.0
    - HTTP/1.1
  - **2nd generation (Binary base)**
    - SPDY 2009
    - HTTP/2
  - **3rd generation (UDP base)**
    - HTTP/3

HTTP Service(Hypertext Transfer Protocol)

- **HTTP/0.9: The one liner (1991)**
  - Support single method “GET” (Ex: GET /index.html)
  - Reply with HTML (data response)
  - Support HTML text only
- **HTTP/1.0: The standard (1996)**
  - Support more method ex: “POST”, “HEAD”
  - Support multiple type of data (image, video, text etc)
  - Add http header both client and server
    - Example
      - Method: xxx
      - Host: xxx
      - User-Agent: xxxx
      - Content-Type: text/plan, text/html, image/jpg etc
  - Add http response code for indicate response to client
  - 1 Connection per request (Poor performance)

HTTP Service(Hypertext Transfer Protocol)

- **HTTP/1.1: The http master (1999)**
  - Support more method “PUT”, “PATCH”, “OPTION”, “DELETE” etc
  - More restrict/security on protocol
    - Restrict “Host” on client’s header
    - Support proxy authentication

  - Persistency connection:
    - Client-Server connection still exist when three-way handshake done
    - Allow client send multi sequential request on same connection
    - Client need to send request “Connection: Close” when all request done

HTTP Service(Hypertext Transfer Protocol)

- **HTTP/1.1: The http master (1999)**
  - Pipeline
    - Add header “Content-Length” for client to indicate when response was end (Static content).
    - Client can send request multiple to server without wait response done one-by-one
  - Chunked Transfer
    - Add header “Transfer-Encoding: chunked” for support dynamic length data transfer
  - Define character set
  - Client cookie
  - Compression
  - Caching
  - etc

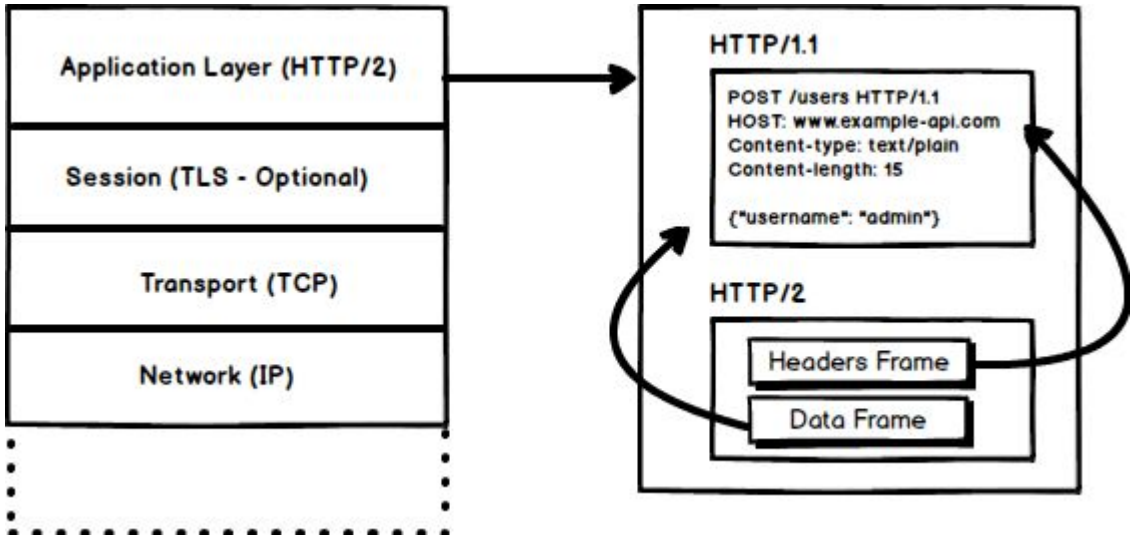
HTTP Service(Hypertext Transfer Protocol)

- **SPDY: HTTP with binary transfer (2009)**
  - More than 10 years (1999 - 2009). HTTP/1.1 is http master.
  - Google was invented and announce as unofficial on 2009
  - SPDY is work on L4 layer with support HTTP (Technically it will modified packet on L4 before send to lower layer). So this not exactly replace HTTP
  - Enhancement on SPDY
    - Priorities on packet
    - Compression for better performance
    - Multiplex multi object send/response on single request
    - Etc
  - After 2015. HTTP/2 was announce (inspired by SPDY). Google was announced deprecated for SPDY

HTTP Service(Hypertext Transfer Protocol)

- **HTTP/2: The HTTP binary master (2015)**
  - Inspired by SPDY (Design for low latency)
  - This new standard for http since 2015 - Now.
  - 6 Element was major change on HTTP/2
    - **Binary protocol**
      - Reduce latency for change all ascii to binary
      - No human readable anymore
      - Divide packet to “Frame”
        - **HEADERS**
        - **DATA**
        - **RST_STREAM ⇒ Client send to brake stream**
        - **SETTINGS**
        - **PRIORITY**
        - **PUSH_PROMISE**
        - **etc**

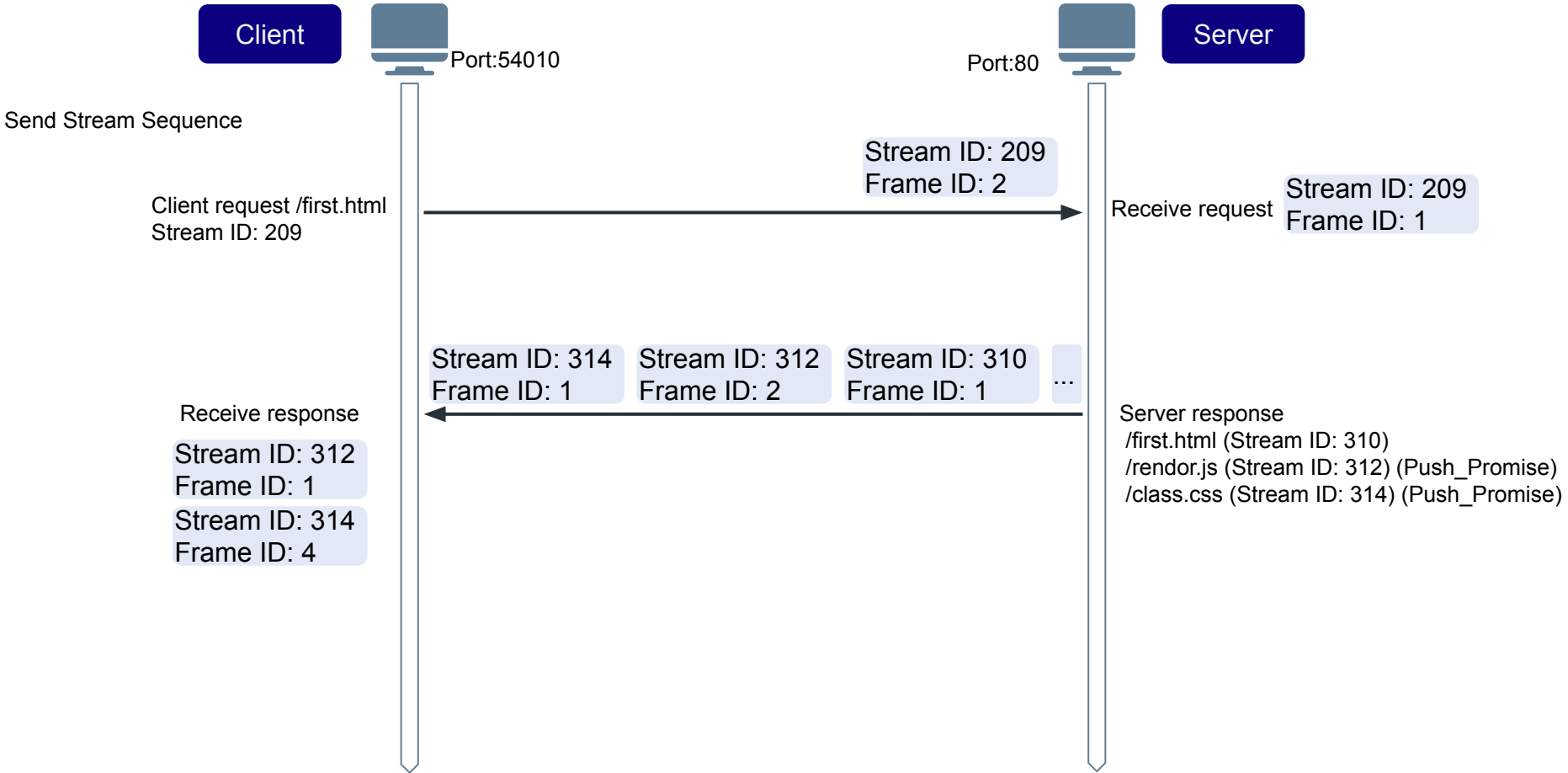
HTTP Service(Hypertext Transfer Protocol)



HTTP Service(Hypertext Transfer Protocol)

- **HTTP/2: The HTTP binary master (2015)**
  - 6 Element was major change on HTTP/2
    - **Binary protocol**
      - **Frame and Stream**
        - All HTTP/2 for request and response will define by unique “Stream ID”
        - All client request will use **odd number** of id, All server response will use **even number (RFC7540)**
      - Each request and response can divide to multiple frames with container their “stream id” and common header of each frame
      - HTTP/2 also have stream id and header in frame for reassembly packet

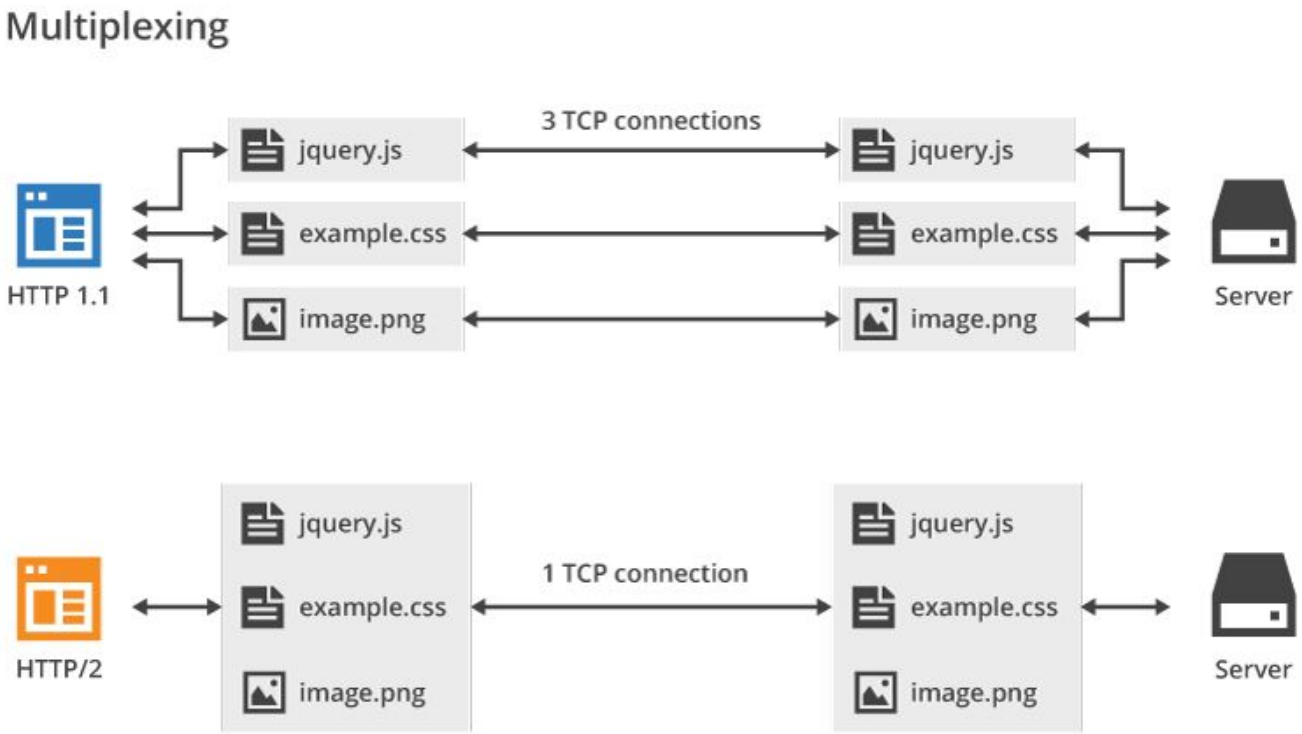
HTTP Service(Hypertext Transfer Protocol)



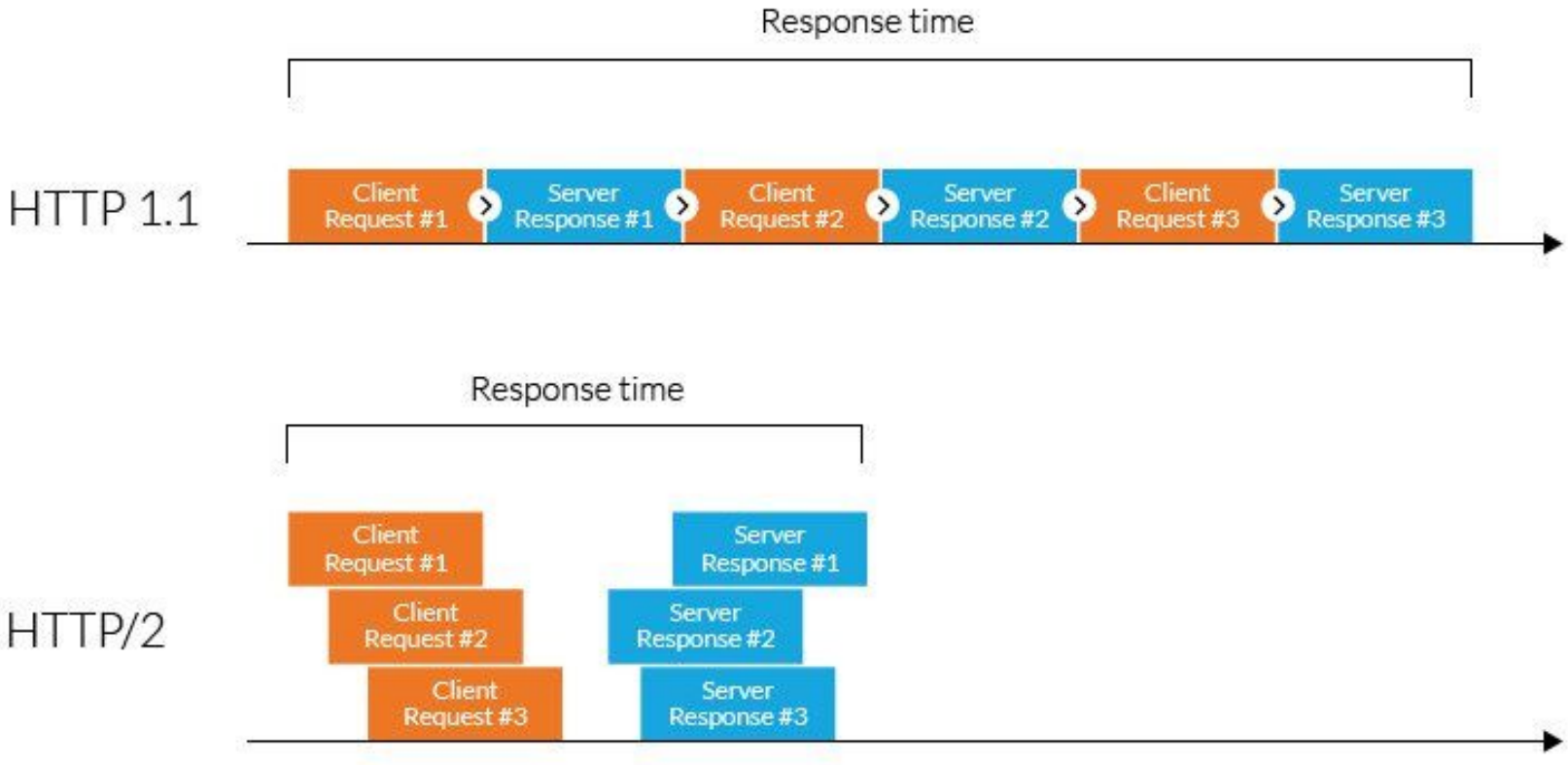
HTTP Service(Hypertext Transfer Protocol)

- **HTTP/2: The HTTP binary master (2015)**
  - 6 Element was major change on HTTP/2
    - **Multiplexing**
      - TCP will establish (three-way handshake) only single connection and allow send/receive stream asynchronous in same connection
      - Fix problem “head-of-line blocking” on tcp level
      - No order, No waiting response (Parallel)
    - **Server Push**
      - When server knowing some elements that client will ask for. They push it to client without ask !!!
      - This process server will use frame type: PUSH_PROMISE
      - This reduce round-trip for request/response between client/server

HTTP Service(Hypertext Transfer Protocol)



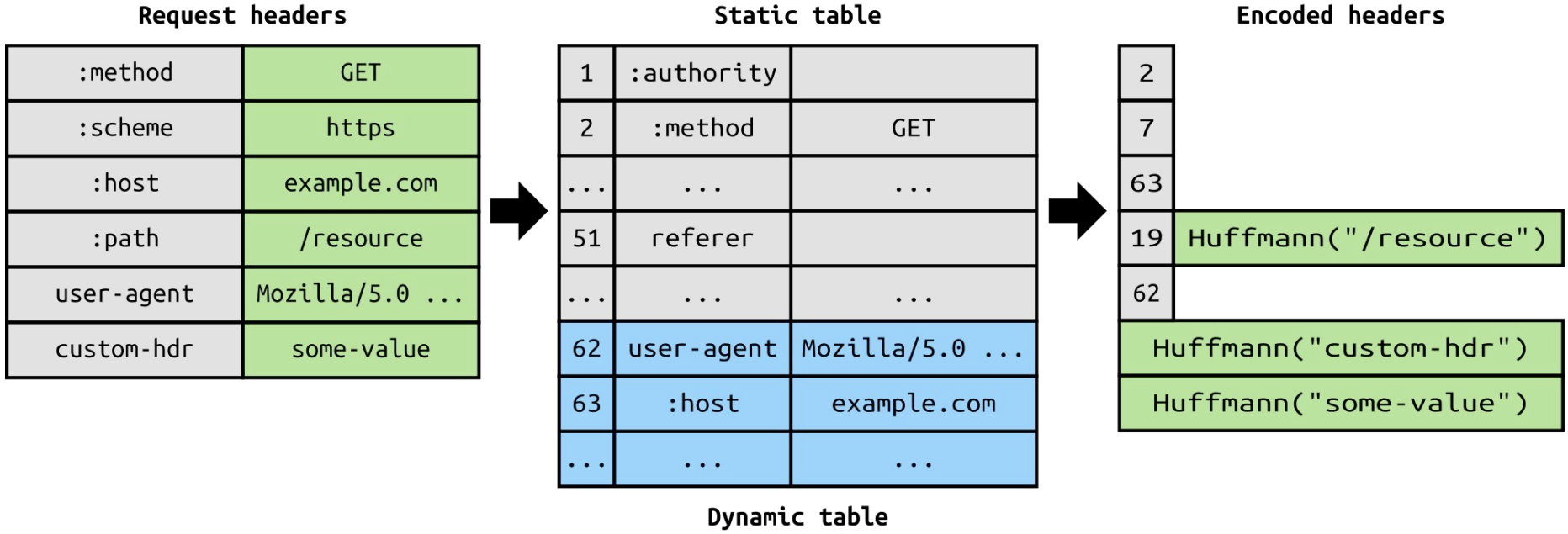
HTTP Service(Hypertext Transfer Protocol)



HTTP Service(Hypertext Transfer Protocol)

- **HTTP/2: The HTTP binary master (2015)**
  - 6 Element was major change on HTTP/2
    - **HPACK Header Compression**
      - All header value will same as HTTP/1.1
      - Optimized send header by compression
      - Encode by Huffman code
      - Client/Server will maintain table of encode header
    - **Request Priority**
      - Client can set priority information on HEADER frame when open stream
      - Anytime client can request to change priority by frame “PRIORITY”
      - Server will use this priority for send response

HTTP Service(Hypertext Transfer Protocol)



HTTP Service(Hypertext Transfer Protocol)

- **HTTP/2: The HTTP binary master (2015)**
  - 6 Element was major change on HTTP/2
    - **HPACK Header Compression**
      - All header value will same as HTTP/1.1
      - Optimized send header by compression
      - Encode by Huffman code
      - Client/Server will maintain table of encode header
    - **Request Priority**
      - Client can set priority information on HEADER frame when open stream
      - Anytime client can request to change priority by frame “PRIORITY”
      - Server will use this priority for send response

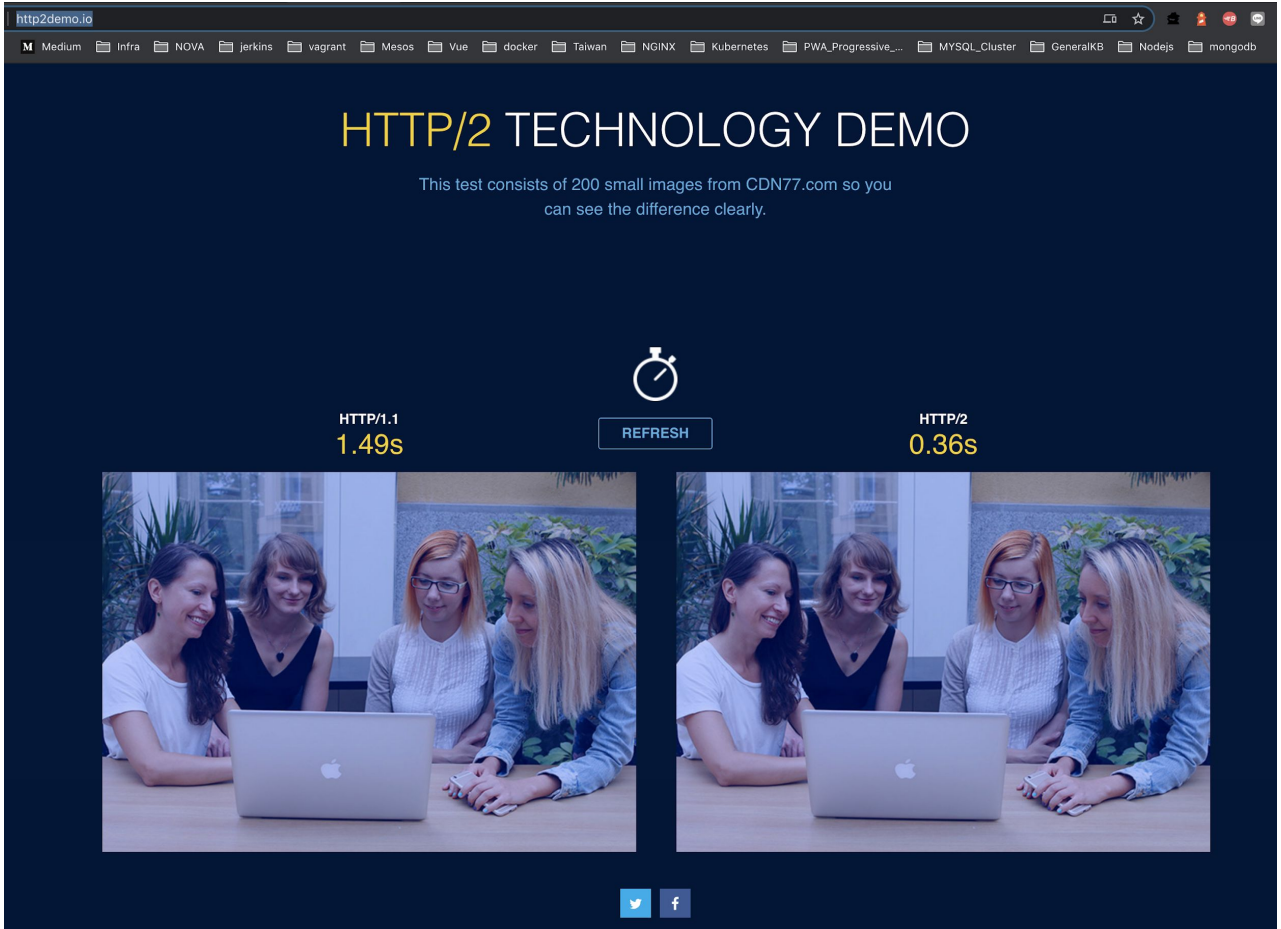
HTTP Service(Hypertext Transfer Protocol)

- **HTTP/2: The HTTP binary master (2015)**
  - 6 Element was major change on HTTP/2
    - **Security**
      - By RFC document not define for secure HTTP/2
      - But as all vendor require TLS over HTTP/2. So this like restriction for operate with this
      - HTTP/2 was TLS (Transport layer security) need to be TLS1.2 or upper
      - So finally HTTP/2 can said that it need to implement with TLS1.2 for encryption that will secure by default.

HTTP Service(Hypertext Transfer Protocol)

```
praparns-MacBook-Pro:~ praparn$ curl -v -k --http2 https://map.google.com
* Rebuilt URL to: https://map.google.com/
*   Trying 172.217.24.174...
* TCP_NODELAY set
* Connected to map.google.com (172.217.24.174) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
*   CAfile: /etc/ssl/cert.pem
  CApath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-ECDSA-CHACHA20-POLY1305
* ALPN, server accepted to use h2
* Server certificate:
*  subject: C=US; ST=California; L=Mountain View; O=Google LLC; CN=*.google.com
*  start date: Nov 10 14:34:43 2020 GMT
*  expire date: Feb  2 14:34:42 2021 GMT
*  issuer: C=US; O=Google Trust Services; CN=GTS CA 1O1
*  SSL certificate verify ok.
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* Using Stream ID: 1 (easy handle 0x7fe913008c00)
> GET / HTTP/2
> Host: map.google.com
> User-Agent: curl/7.54.0
> Accept: */*
>
* Connection state changed (MAX_CONCURRENT_STREAMS updated)!
< HTTP/2 301
< location: https://maps.google.com/
< content-type: text/html; charset=UTF-8
< x-content-type-options: nosniff
< date: Sun, 13 Dec 2020 16:05:21 GMT
< expires: Sun, 13 Dec 2020 16:35:21 GMT
< server: sffe
< content-length: 221
< x-xss-protection: 0
< cache-control: public, max-age=1800
< age: 689
< alt-svc: h3-29=":443"; ma=2592000,h3-T051=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"
<
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="https://maps.google.com/">here</A>.
</BODY></HTML>
* Connection #0 to host map.google.com left intact
praparns-MacBook-Pro:~ praparn$
```

HTTP Service(Hypertext Transfer Protocol)



HTTP Service(Hypertext Transfer Protocol)

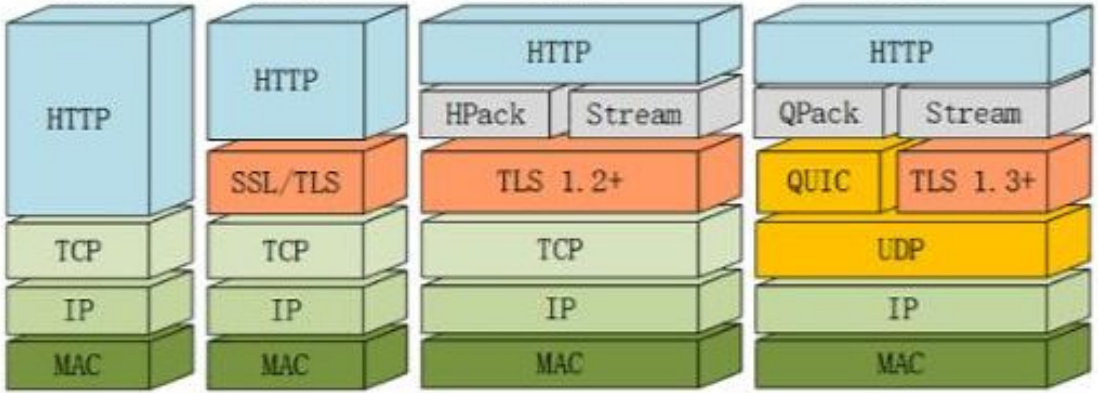
- **HTTP/3: The next chapter of HTTP on UDP (2018)**
  - Initially by google again, Still draft status on IEEE
  - Many name in evolution of this standard
    - HTTP/2 Semantics Using The QUIC Transport Protocol
    - Hypertext Transfer Protocol (HTTP) over QUIC
    - HTTP/3 (Final on 28 Oct 2018)
  - Major change from HTTP/1.1 and HTTP/2 by change connection from tcp to udp (Improvement on this will increase performance
  - Fixing problem “head-of-line blocking” on tcp level
  - Zero RTT (round trip time) for establish connection
  - All major browser are announce to support http/3 standard
    - Edge
    - Chrome
    - FireFox
    - Safari

HTTP Service(Hypertext Transfer Protocol)

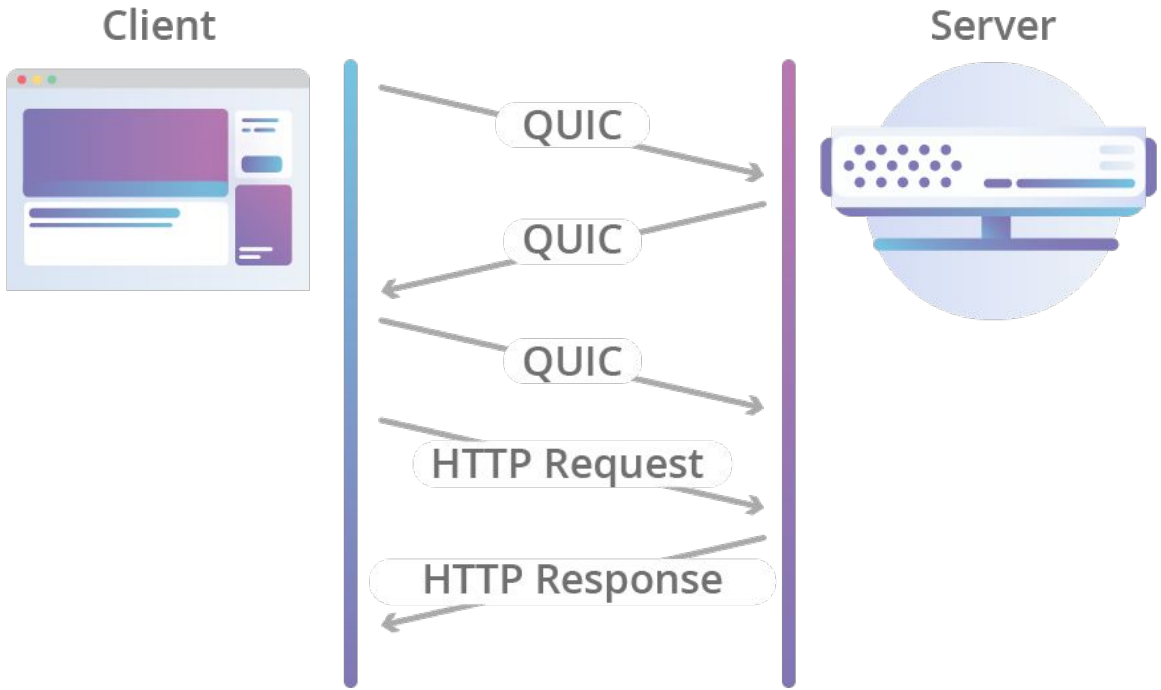
Browser	Version implemented (disabled by default)		Version shipped (enabled by default)	
Chrome	Stable build (79)	December 2019		TBA
Firefox	Stable build (72.0.1)	January 2020		TBA
Safari	Safari Technology Preview 104	April 2020	Safari 14	September 2020
Edge	Edge (Canary build)	May 2020[13]		TBA

HTTP Service(Hypertext Transfer Protocol)

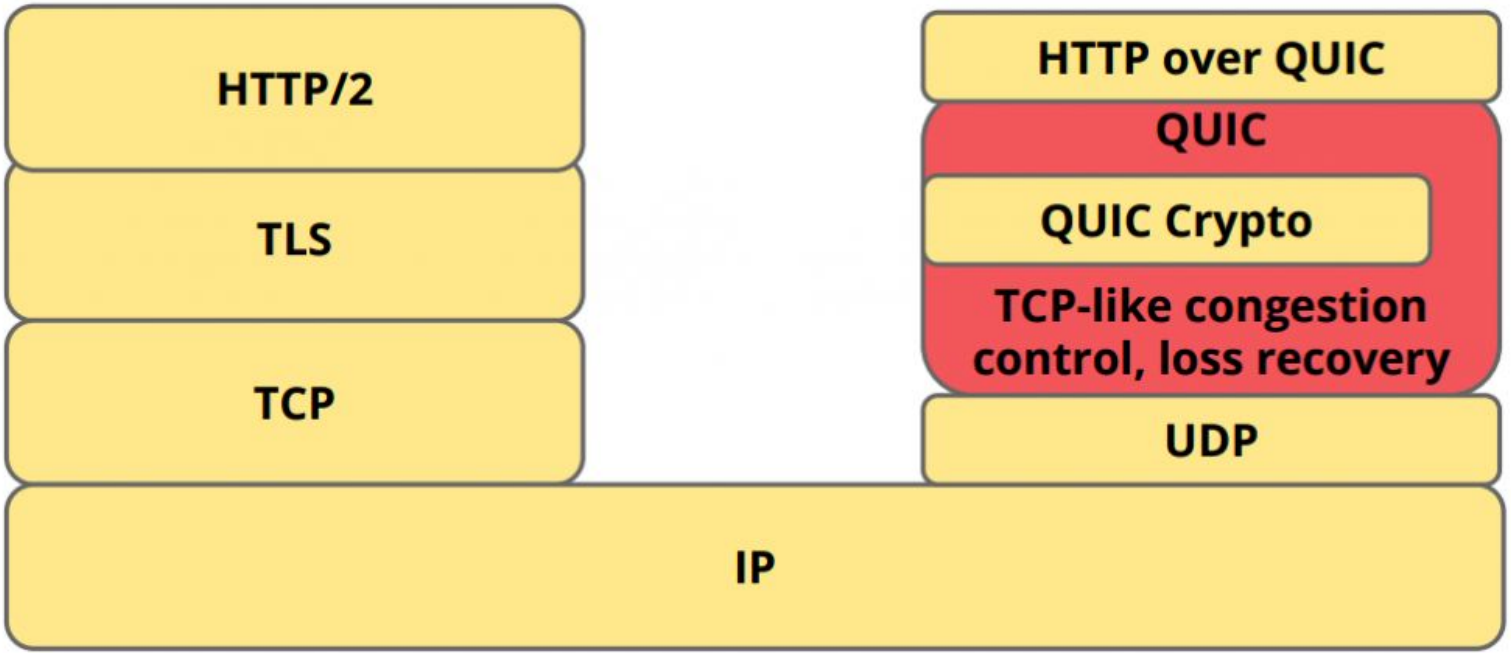
- **HTTP/3: The next chapter of HTTP on UDP (2018)**
  - HTTP/3 also have three-way handshake with QUIC
  - This not only evolution of HTTP/2 but whole new HTTP !!!
  - HPACK will change to QPACK
  - Secure by default with TLS 1.3+



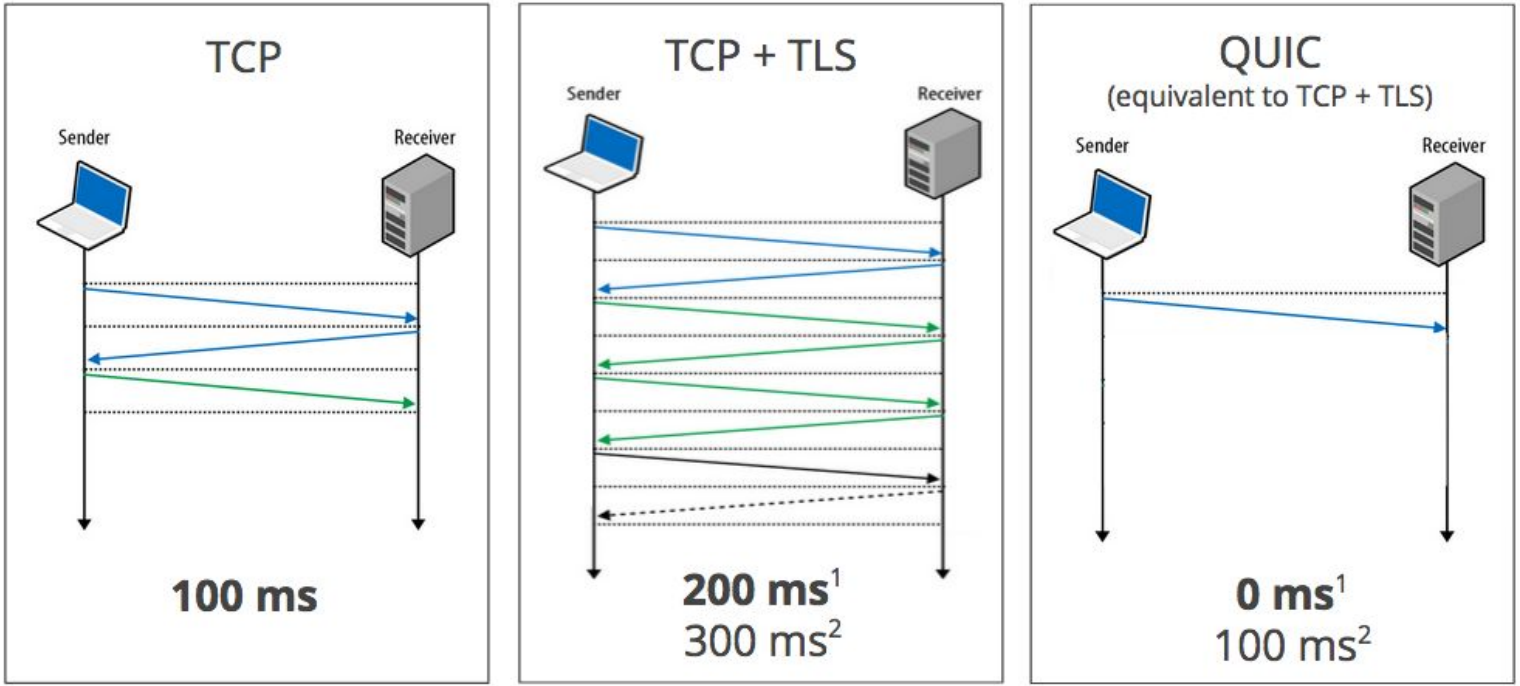
HTTP Service(Hypertext Transfer Protocol)



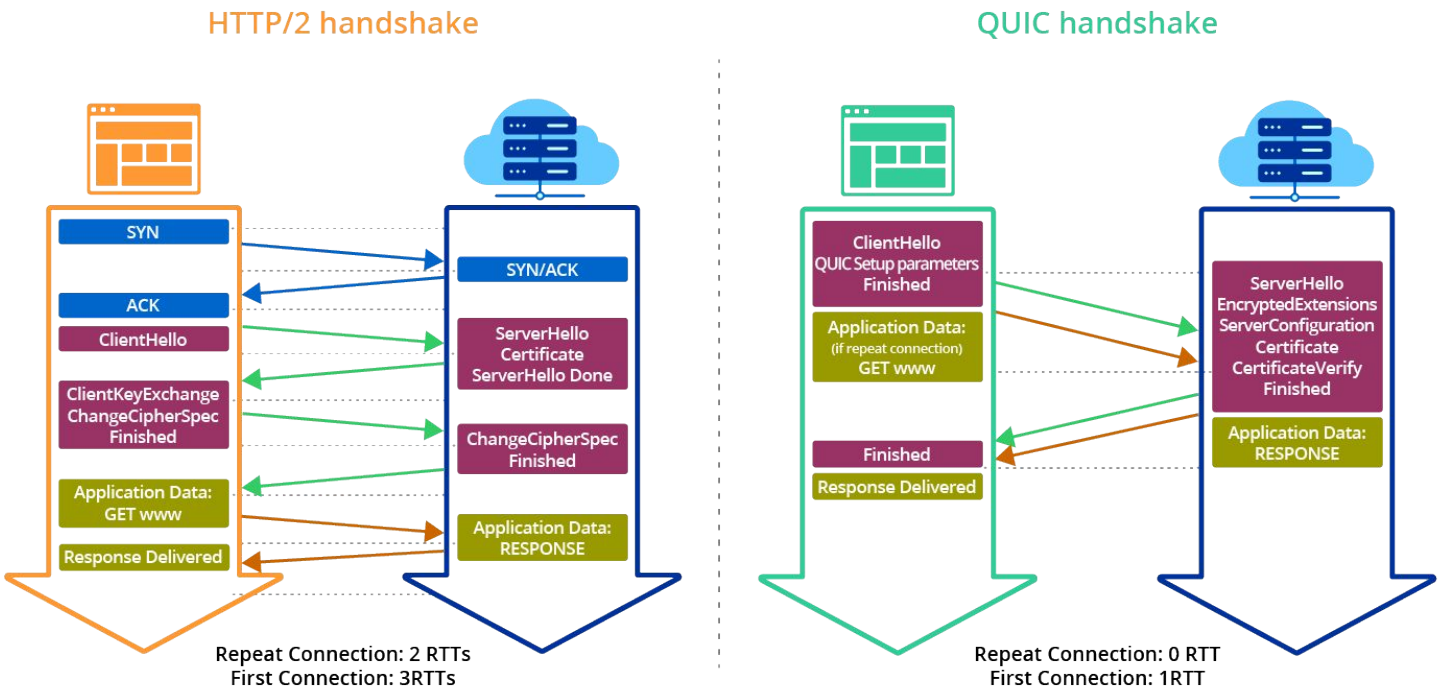
HTTP Service(Hypertext Transfer Protocol)



HTTP Service(Hypertext Transfer Protocol)



HTTP Service(Hypertext Transfer Protocol)



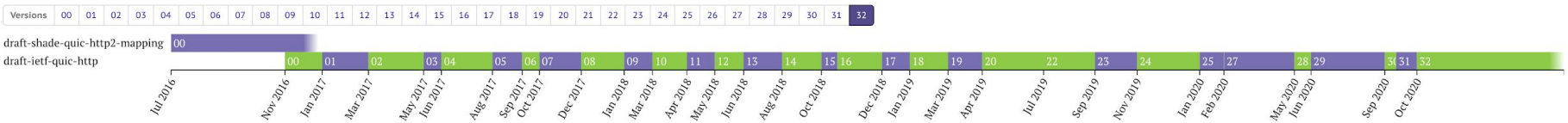
HTTP Service(Hypertext Transfer Protocol)

IETF Datatracker Groups Documents Meetings Other User Document search

Hypertext Transfer Protocol Version 3 (HTTP/3)

draft-ietf-quic-http-32

Status | IESG evaluation record | IESG writeups | Email expansions | History



Document	**Type**	Active Internet-Draft (quic WG)
	Author	Mike Bishop
	Last updated	2020-11-16 (latest revision 2020-10-20)
	Replaces	draft-shade-quic-http2-mapping
	Stream	IETF
	Intended RFC status	Proposed Standard
	Formats	plain text html xml pdf htmlized (tools) htmlized bibtex
	Reviews	SECDIR Last Call Review: Ready GENART Last Call Review: Ready with Issues *OPSDIR Last Call Review - due: 2020-11-16*
Stream	**WG state**	Submitted to IESG for Publication (*wg milestone: Jul 2020 - HTTP/2 mapping docum...*)
	Document shepherd	Lucas Pardue
	Shepherd write-up	Show (last changed 2020-09-25)

SSL/TLS

- **SSL/TLS not mean always for HTTPS !!!**
  - SSL/TLS is cryptographic protocol for secure communication over network
  - SSL all version was announce to deprecated (SSL V2.0, V3.0)
  - TLS version 1.0 and 1.1 also got announce to deprecate
  - Only TLS 1.2 and TLS 1.3 is now production
  - Many protocol will use SSL/TLS for secure connection
    - Mail (SMTP) + TLS ⇒ SMTPS
    - Web (HTTP) + TLS ⇒ HTTPS
    - Telnet + TLS ⇒ Telnet over SSL
    - FTP + TLS ⇒ SFTP

SSL/TLS

Protocol	Publish	Creator	Remarks
SSL 1.0	Unpublished	Netscape	Unpublished
SSL 2.0	1995	Netscape	Deprecated 2011
SSL 3.0	1996	Netscape ⇒ IETF	Deprecated 2015
TLS 1.0/SSL 3.1	1999	IETF	Deprecated 2020
TLS 1.1	2006	IETF	Deprecated 2020
TLS 1.2	2008	IETF	
TLS 1.3	2018	IETF	

SSL/TLS Cipher

- **TLS cipher suite**
  - Set of algorithm for helper client/server negotiate encryption over TLS on same suite
  - Cipher suite was announced as standard for each version of TLS
  - Issue about incompatible of cipher suite is usually case for use TLS for secure protocol
  - Cipher suite have naming conversion for identify as example below (IANA Format)
  - “ TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384”

“Protocol”
TLS is default

“Key Exchange Algorithm”
Determine how client/server exchange on handshake phase

“Authentication Algorithm”
Determine how client/server authentication on handshake phase

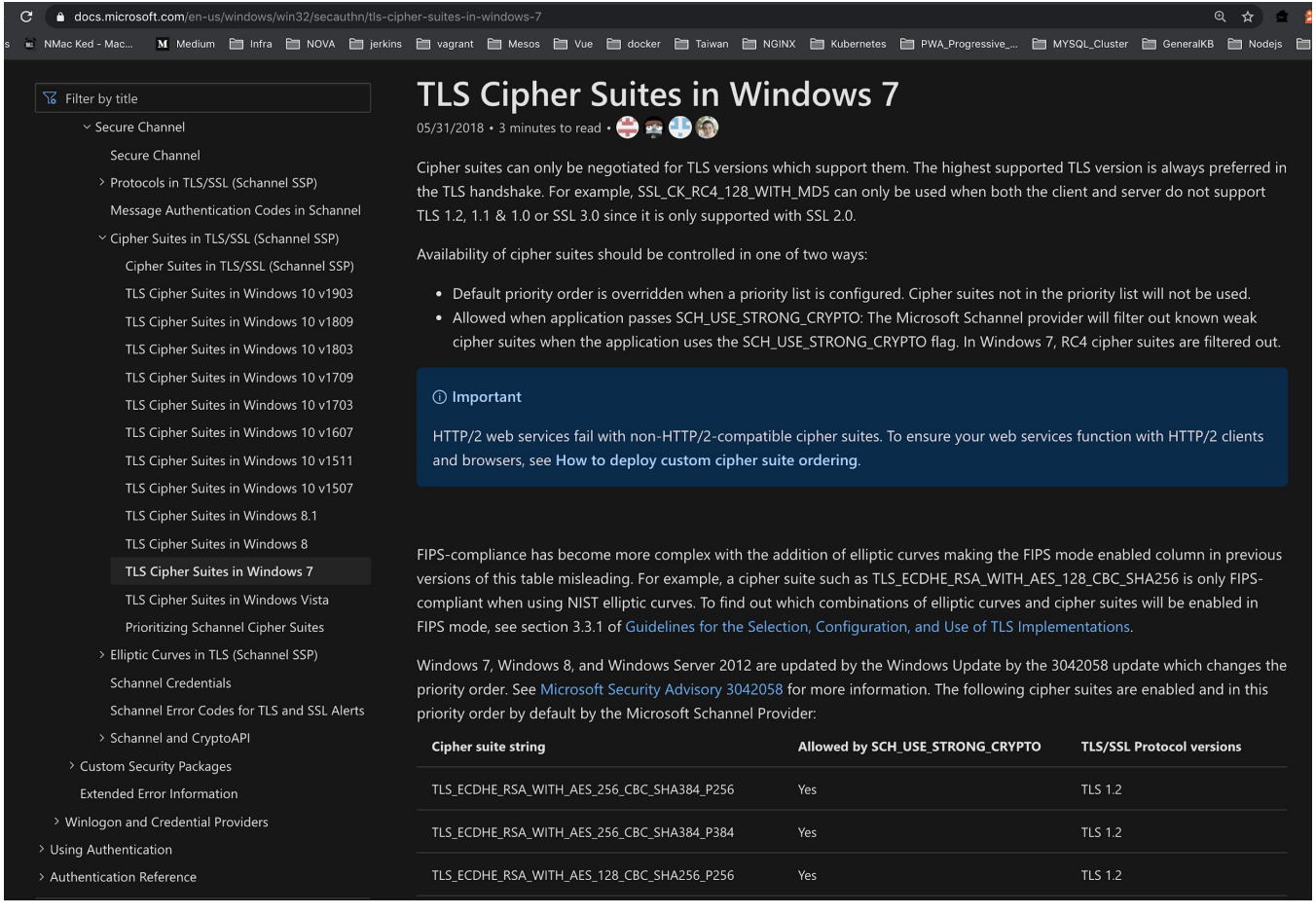
“Bulk Encryption Algorithm”
Use encrypt data on TLS

“Hash algorithm”
Hash for check data integrity

SSL/TLS Cipher

	Akamai/OpenSSL cipher names	ak-akamai-2020q1 (Default)	ak-akamai-2016q3	ak-akamai-default-2017q3	ak-pci-dss-3.2	ak-akamai-default-2016q3	ak-akamai-default-2016q1	ak-akamai-pfs-supported	ak-akamai-default	ak-pci-dss-3.1	TLSv1.3	TLSv1.2	TLSv1.1	TLSv1	RSA Certificate	ECDSA Certificate	IANA cipher names	Akamai cipher code	OpenSSL Hex Code
		Current profiles			Existing certificates only					EOL	TLS versions				Signature		IANA cipher names		
Forward Secrecy ciphers	TLS-AES-256-GCM-SHA384	✓	✓	✓							✓				✓	✓	TLS_AES_256_GCM_SHA384	3001302	0x13,0x02
	TLS-CHACHA20-POLY1305-SHA256	✓	✓	✓							✓				✓	✓	TLS_CHACHA20_POLY1305_SHA256	3001303	0x13,0x03
	TLS-AES-128-GCM-SHA256	✓	✓	✓							✓				✓	✓	TLS_AES_128_GCM_SHA256	3001301	0x13,0x01
	TLS-AES-128-CCM-8-SHA256		✓	✓							✓				✓	✓	TLS_AES_128_CCM_8_SHA256	3001305	0x13,0x05
	TLS-AES-128-CCM-SHA256		✓	✓							✓				✓	✓	TLS_AES_128_CCM_SHA256	3001304	0x13,0x04
	ECDHE-ECDSA-AES256-GCM-SHA384	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓				✓	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	300C02C	0xC0,0x2C
	ECDHE-ECDSA-AES128-GCM-SHA256	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓				✓	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	300C02B	0xC0,0x2B
	ECDHE-RSA-AES256-GCM-SHA384	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓			✓		TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	300C030	0xC0,0x30
	ECDHE-RSA-AES128-GCM-SHA256	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓			✓		TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	300C02F	0xC0,0x2F
	ECDHE-ECDSA-CHACHA20-POLY1305	✓	✓	✓		✓	✓					✓				✓	TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	300CCA9	0xCC,0xA9
	ECDHE-RSA-CHACHA20-POLY1305	✓	✓	✓		✓	✓					✓			✓		TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	300CCA8	0xCC,0xA8
	ECDHE-ECDSA-AES256-SHA384		✓	✓	✓	✓	✓	✓	✓	✓		✓				✓	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	300C024	0xC0,0x24
	ECDHE-ECDSA-AES128-SHA256		✓	✓	✓	✓	✓	✓	✓	✓		✓				✓	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	300C023	0xC0,0x23
	ECDHE-RSA-AES256-SHA384		✓	✓	✓	✓	✓	✓	✓	✓		✓			✓		TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	300C028	0xC0,0x28
	ECDHE-RSA-AES128-SHA256		✓	✓	✓	✓	✓	✓	✓	✓		✓			✓		TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	300C027	0xC0,0x27
	ECDHE-RSA-AES256-SHA		✓	✓	✓	✓	✓	✓		✓		✓	✓	✓	✓		TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	300C014	0xC0,0x14
	ECDHE-RSA-AES128-SHA		✓	✓	✓	✓	✓	✓		✓		✓	✓	✓	✓		TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	300C013	0xC0,0x13
Non-PFS Ciphers	AES256-GCM-SHA384			✓	✓	✓	✓	✓	✓	✓		✓			✓		TLS_RSA_WITH_AES_256_GCM_SHA384	300009D	0x00,0x9D
	AES128-GCM-SHA256			✓	✓	✓	✓	✓	✓	✓		✓			✓		TLS_RSA_WITH_AES_128_GCM_SHA256	300009C	0x00,0x9C
	AES256-SHA256			✓	✓	✓	✓	✓	✓	✓		✓			✓		TLS_RSA_WITH_AES_256_CBC_SHA256	300003D	0x00,0x3D
	AES128-SHA256			✓	✓	✓	✓	✓	✓	✓		✓			✓		TLS_RSA_WITH_AES_128_CBC_SHA256	300003C	0x00,0x3C
	AES256-SHA			✓	✓	✓	✓			✓		✓	✓	✓	✓		TLS_RSA_WITH_AES_256_CBC_SHA	3000035	0x00,0x35
	AES128-SHA			✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		TLS_RSA_WITH_AES_128_CBC_SHA	300002F	0x00,0x2F
	RC4-SHA								✓			✓	✓	✓	✓		TLS_RSA_WITH_RC4_128_SHA	3000005	0x00,0x05
	RC4-MD5											✓	✓	✓	✓		TLS_RSA_WITH_RC4_128_MD5	3000004	0x00,0x04
	DES-CBC3-SHA						✓	✓	✓	✓		✓	✓	✓	✓		TLS_RSA_WITH_3DES_EDE_CBC_SHA	300000A	0x00,0x0A

SSL/TLS Cipher



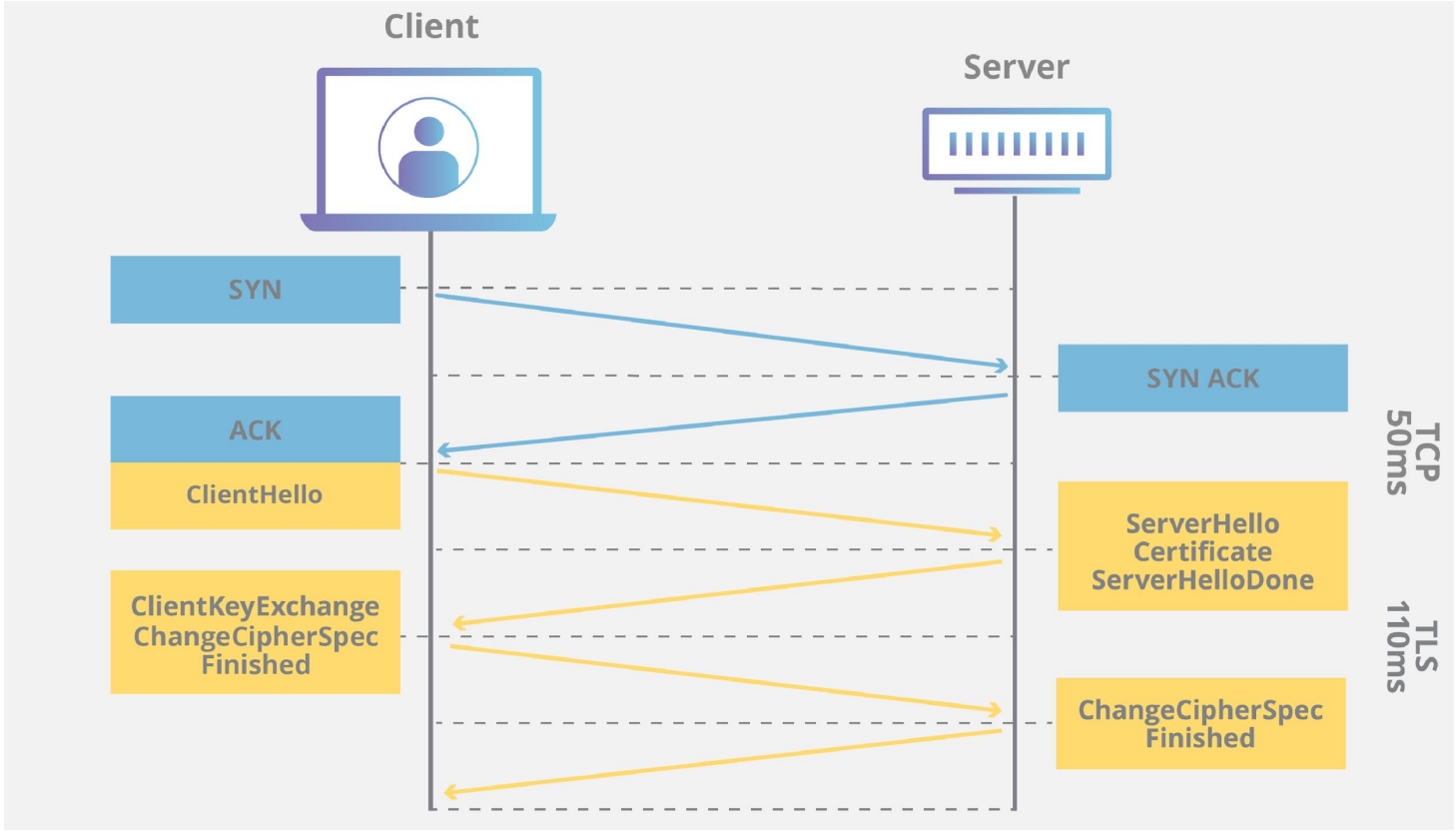
SSL/TLS Flow Control

- **TLS with RSA Exchange Algorithm (RSA):**
  - **Client Hello**: Client establish tls to server with tls version, cipher suite, random no (client random)
  - **Server Hello**: Server reply with tls version, cipher suite, random no (server random) and server certificate (public key)
  - **Authentication:** Client verify certificate with CA
  - **Pre-master secret:** Client generate random no (pre-master) and encrypt with public key send to server
  - **Decrypt pre-master**: Server decrypt pre-master with private key
  - **Establish session-key**: Client & Server generate session-key from pre-master, client random and server random (will be same)
  - **Client ready**: Client send message “finished” with encrypt by session-key to server for test decrypt
  - **Server ready**: Server send message “finished” with encrypt by session-key to client fot test decrypt
  - **Handshake complete**: Start communication

SSL/TLS Flow Control

- **TLS with ephemeral Diffie Hellman Exchange(DHE):**
  - **Client Hello**: Client establish tls to server with tls version, cipher suite, random no (client random)
  - **Server Hello**: Server reply with tls version, cipher suite, random no (server random) and server certificate (public key)
  - **Server's digital signature:** Server encrypt client random, server random and server DH* parameter with private key and send to client
  - **Digital signature confirm:** Client decrypt server's digital signature with public key and send client DH* parameter back
  - **Generate pre-master**: **Client & Server** generate pre-master by DH*
  - **Establish session-key**: Client & Server generate session-key from pre-master, client random and server random (will be same)
  - **Client ready**: Client send message "finished" with encrypt by session-key to server for test decrypt
  - **Server ready**: Server send message "finished" with encrypt by session-key to client fot test decrypt
  - **Handshake complete**: Start communication

SSL/TLS Flow Control



SSL/TLS Flow Control

```
praparns-MacBook-Pro:~ praparn$ curl https://www.google.com -I -v -k --http2
* Rebuilt URL to: https://www.google.com/
*   Trying 172.217.194.104...
* TCP_NODELAY set
* Connected to www.google.com (172.217.194.104) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
*   CAfile: /etc/ssl/cert.pem
  CApath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-ECDSA-CHACHA20-POLY1305
* ALPN, server accepted to use h2
* Server certificate:
*  subject: C=US; ST=California; L=Mountain View; O=Google LLC; CN=www.google.com
*  start date: Nov 10 14:41:09 2020 GMT
*  expire date: Feb  2 14:41:08 2021 GMT
*  issuer: C=US; O=Google Trust Services; CN=GTS CA 1O1
*  SSL certificate verify ok.
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* Using Stream ID: 1 (easy handle 0x7fc41e80a600)
> HEAD / HTTP/2
> Host: www.google.com
> User-Agent: curl/7.54.0
> Accept: */*
>
* Connection state changed (MAX_CONCURRENT_STREAMS updated)!
< HTTP/2 200
HTTP/2 200
< content-type: text/html; charset=ISO-8859-1
content-type: text/html; charset=ISO-8859-1
< p3p: CP="This is not a P3P policy! See g.co/p3phelp for more info."
p3p: CP="This is not a P3P policy! See g.co/p3phelp for more info."
< date: Mon, 14 Dec 2020 17:02:16 GMT
date: Mon, 14 Dec 2020 17:02:16 GMT
< server: gws
server: gws
< x-xss-protection: 0
x-xss-protection: 0
< x-frame-options: SAMEORIGIN
x-frame-options: SAMEORIGIN
< expires: Mon, 14 Dec 2020 17:02:16 GMT
expires: Mon, 14 Dec 2020 17:02:16 GMT
< cache-control: private
cache-control: private
< set-cookie: 1P_JAR=2020-12-14-17; expires=Wed, 13-Jan-2021 17:02:16 GMT; path=/; domain=.google.com; Secure
set-cookie: 1P_JAR=2020-12-14-17; expires=Wed, 13-Jan-2021 17:02:16 GMT; path=/; domain=.google.com; Secure
< set-cookie: NID=204=GemYbfhvlhVMdfk6gryoifgTENiwhG3sD3QZj4esYK5sJ2jR5JRjEUgr3ARoaipr227i4zDUJsPbXrCGOI9-iKen0PMBMG_I0Oi-xwkQodmYoihSAN-9mD7exlXtXM0ymRxeI9r_msGjRu91zv4gYme2ko
GA5Nynh_YruJjD0NQ; expires=Tue, 15-Jun-2021 17:02:16 GMT; path=/; domain=.google.com; HttpOnly
set-cookie: NID=204=GemYbfhvlhVMdfk6gryoifgTENiwhG3sD3QZj4esYK5sJ2jR5JRjEUgr3ARoaipr227i4zDUJsPbXrCGOI9-iKen0PMBMG_I0Oi-xwkQodmYoihSAN-9mD7exlXtXM0ymRxeI9r_msGjRu91zv4gYme2koGA
5Nynh_YruJjD0NQ; expires=Tue, 15-Jun-2021 17:02:16 GMT; path=/; domain=.google.com; HttpOnly
< alt-svc: h3-29=":443"; ma=2592000,h3-T051=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,4
3"
alt-svc: h3-29=":443"; ma=2592000,h3-T051=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"

<
* Connection #0 to host www.google.com left intact
praparns-MacBook-Pro:~ praparn$
```

SSL/TLS Flow Control

```
praparns-MacBook-Pro:~ praparn$ chmod +x ./cipher.sh
praparns-MacBook-Pro:~ praparn$ more ./cipher.sh
#!/usr/bin/env bash

# OpenSSL requires the port number.
SERVER=$1
DELAY=1
ciphers=$(openssl ciphers 'ALL:eNULL' | sed -e 's/:/ /g')

echo Obtaining cipher list from $(openssl version).

for cipher in ${ciphers[@]}
do
echo -n Testing $cipher...
result=$(echo -n | openssl s_client -cipher "$cipher" -connect $SERVER 2>&1)
if [[ "$result" =~ ":error:" ]] ; then
  error=$(echo -n $result | cut -d':' -f6)
  echo NO \($error\)
else
  if [[ "$result" =~ "Cipher is ${cipher}" || "$result" =~ "Cipher    :" ]] ; then
    echo YES
  else
    echo UNKNOWN RESPONSE
    echo $result
  fi
fi
sleep $DELAY
done
praparns-MacBook-Pro:~ praparn$
```

```
praparns-MacBook-Pro:~ praparn$ ./cipher.sh www.google.com:443
Obtaining cipher list from LibreSSL 2.6.5.
Testing ECDHE-RSA-AES256-GCM-SHA384...YES
Testing ECDHE-ECDSA-AES256-GCM-SHA384...NO (sslv3 alert handshake failure)
Testing ECDHE-RSA-AES256-SHA384...NO (sslv3 alert handshake failure)
Testing ECDHE-ECDSA-AES256-SHA384...NO (sslv3 alert handshake failure)
Testing ECDHE-RSA-AES256-SHA...YES
Testing ECDHE-ECDSA-AES256-SHA...NO (sslv3 alert handshake failure)
Testing DHE-RSA-AES256-GCM-SHA384...NO (sslv3 alert handshake failure)
Testing DHE-RSA-AES256-SHA256...NO (sslv3 alert handshake failure)
Testing DHE-RSA-AES256-SHA...NO (sslv3 alert handshake failure)
Testing ECDHE-ECDSA-CHACHA20-POLY1305...NO (sslv3 alert handshake failure)
Testing ECDHE-RSA-CHACHA20-POLY1305...YES
Testing DHE-RSA-CHACHA20-POLY1305...NO (sslv3 alert handshake failure)
Testing GOST2012256-GOST89-GOST89...NO (sslv3 alert handshake failure)
Testing DHE-RSA-CAMELLIA256-SHA256...NO (sslv3 alert handshake failure)
Testing DHE-RSA-CAMELLIA256-SHA...NO (sslv3 alert handshake failure)
Testing GOST2001-GOST89-GOST89...NO (sslv3 alert handshake failure)
Testing AECDH-AES256-SHA...NO (sslv3 alert handshake failure)
Testing ADH-AES256-GCM-SHA384...NO (sslv3 alert handshake failure)
Testing ADH-AES256-SHA256...NO (sslv3 alert handshake failure)
Testing ADH-AES256-SHA...NO (sslv3 alert handshake failure)
Testing ADH-CAMELLIA256-SHA256...NO (sslv3 alert handshake failure)
Testing ADH-CAMELLIA256-SHA...NO (sslv3 alert handshake failure)
Testing AES256-GCM-SHA384...YES
Testing AES256-SHA256...NO (sslv3 alert handshake failure)
Testing AES256-SHA...YES
Testing CAMELLIA256-SHA256...NO (sslv3 alert handshake failure)
Testing CAMELLIA256-SHA...NO (sslv3 alert handshake failure)
Testing ECDHE-RSA-AES128-GCM-SHA256...YES
Testing ECDHE-ECDSA-AES128-GCM-SHA256...NO (sslv3 alert handshake failure)
Testing ECDHE-RSA-AES128-SHA256...NO (sslv3 alert handshake failure)
Testing ECDHE-ECDSA-AES128-SHA256...NO (sslv3 alert handshake failure)
Testing ECDHE-RSA-AES128-SHA...YES
Testing ECDHE-ECDSA-AES128-SHA...NO (sslv3 alert handshake failure)
Testing DHE-RSA-AES128-GCM-SHA256...NO (sslv3 alert handshake failure)
Testing DHE-RSA-AES128-SHA256...NO (sslv3 alert handshake failure)
Testing DHE-RSA-AES128-SHA...NO (sslv3 alert handshake failure)
Testing DHE-RSA-CAMELLIA128-SHA256...NO (sslv3 alert handshake failure)
Testing DHE-RSA-CAMELLIA128-SHA...NO (sslv3 alert handshake failure)
Testing AECDH-AES128-SHA...NO (sslv3 alert handshake failure)
Testing ADH-AES128-GCM-SHA256...NO (sslv3 alert handshake failure)
Testing ADH-AES128-SHA256...NO (sslv3 alert handshake failure)
Testing ADH-AES128-SHA...NO (sslv3 alert handshake failure)
Testing ADH-CAMELLIA128-SHA256...NO (sslv3 alert handshake failure)
Testing ADH-CAMELLIA128-SHA...NO (sslv3 alert handshake failure)
Testing AES128-GCM-SHA256...YES
Testing AES128-SHA256...NO (sslv3 alert handshake failure)
```

Workshop 2.2 HTTP/TLS Service

```
ubuntu@ip-10-21-2-22:~$ curl https://www.google.com -v --tlsv1.2 -L -I
* Rebuilt URL to: https://www.google.com/
*   Trying 172.217.194.103...
* TCP_NODELAY set
* Connected to www.google.com (172.217.194.103) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: /etc/ssl/certs/ca-certificates.crt
  CApath: /etc/ssl/certs
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-ECDSA-CHACHA20-POLY1305
* ALPN, server accepted to use h2
* Server certificate:
*  subject: C=US; ST=California; L=Mountain View; O=Google LLC; CN=www.google.com
*  start date: Nov 10 14:41:09 2020 GMT
*  expire date: Feb  2 14:41:08 2021 GMT
*  subjectAltName: host "www.google.com" matched cert's "www.google.com"
*  issuer: C=US; O=Google Trust Services; CN=GTS CA 1O1
*  SSL certificate verify ok.
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* Using Stream ID: 1 (easy handle 0x556dee3c25c0)
> HEAD / HTTP/2
> Host: www.google.com
> User-Agent: curl/7.58.0
> Accept: */*
>
* Connection state changed (MAX_CONCURRENT_STREAMS updated)!
< HTTP/2 200
HTTP/2 200
< content-type: text/html; charset=ISO-8859-1
content-type: text/html; charset=ISO-8859-1
< p3p: CP="This is not a P3P policy! See g.co/p3phelp for more info."
p3p: CP="This is not a P3P policy! See g.co/p3phelp for more info."
< date: Sun, 27 Dec 2020 07:59:02 GMT
date: Sun, 27 Dec 2020 07:59:02 GMT
< server: gws
server: gws
< x-xss-protection: 0
x-xss-protection: 0
< x-frame-options: SAMEORIGIN
x-frame-options: SAMEORIGIN
< expires: Sun, 27 Dec 2020 07:59:02 GMT
expires: Sun, 27 Dec 2020 07:59:02 GMT
< cache-control: private
cache-control: private
< set-cookie: 1P_JAR=2020-12-27-07; expires=Tue, 26-Jan-2021 07:59:02 GMT; path=/; domain=.google.com; Secure
set-cookie: 1P_JAR=2020-12-27-07; expires=Tue, 26-Jan-2021 07:59:02 GMT; path=/; domain=.google.com; Secure
< set-cookie: NID=205=kPp5QAtRx-VbuwFYLVpnel-0UGbP4etl_bVjniBRh5Kf4lvsXnhfwGj3XzM5fc-wp94scdFVO2xGKvrFTEXT0dUOzw2kNPBwf1TLU8e4RY5y1CZOJPsbhjZ6VJjRoTQy97GWJp7IyYsmNd3JGD8F19ZyC2p548PtWpWfKLDRSR8; expires=Mon, 28-Jun-2021 07:59:02 GMT; path=/; domain=.google.com; HttpOnly
set-cookie: NID=205=kPp5QAtRx-VbuwFYLVpnel-0UGbP4etl_bVjniBRh5Kf4lvsXnhfwGj3XzM5fc-wp94scdFVO2xGKvrFTEXT0dUOzw2kNPBwf1TLU8e4RY5y1CZOJPsbhjZ6VJjRoTQy97GWJp7IyYsmNd3JGD8F19ZyC2p548PtWpWfKLDRSR8; expires=Mon, 28-Jun-2021 07:59:02 GMT; path=/; domain=.google.com; HttpOnly
< alt-svc: h3-29=":443"; ma=2592000,h3-T051=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"
alt-svc: h3-29=":443"; ma=2592000,h3-T051=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"

<
* Connection #0 to host www.google.com left intact
ubuntu@ip-10-21-2-22:~$
```

WebSocket

- **When HTTP is not enough!!!**
  - When application need full-duplex connection between client/server, WebSocket is the solution for real-time application
  - WebSocket is distinct from HTTP but still on TCP-80, TCP-443 and initial connection via HTTP method (**HTTP/1.1**)
  - Initial design and start well-know on HTML5 and Chat /video streaming / live feed / multiplayer gaming etc.
  - Many year non-standard until RFC 6455 on 2011
  - Minimize process for fast communication handshake
  - As bi-directional this no more round-trip connection (No request/response)
  - Modern browser all support WebSocket
    - Chrome 16+
    - Firefox 11+
    - Opera 12.1+

WebSocket

Comparison ViewPoint	HTTP Protocol	WebSocket Protocol
Duplex	Half	Full Duplex
Message Pattern	Request - Response	Bi-directional
Service Push	Client/Server + Server Push	Bi-directional
Data type	Binary (HTTP/2)	Binary or Text
Caching	Core feature	Not possible
Multiplexing	Yes	Yes
Priority	Yes	No
Compression	HPAC (HTTP/2)	No
Retransmit	Yes	No

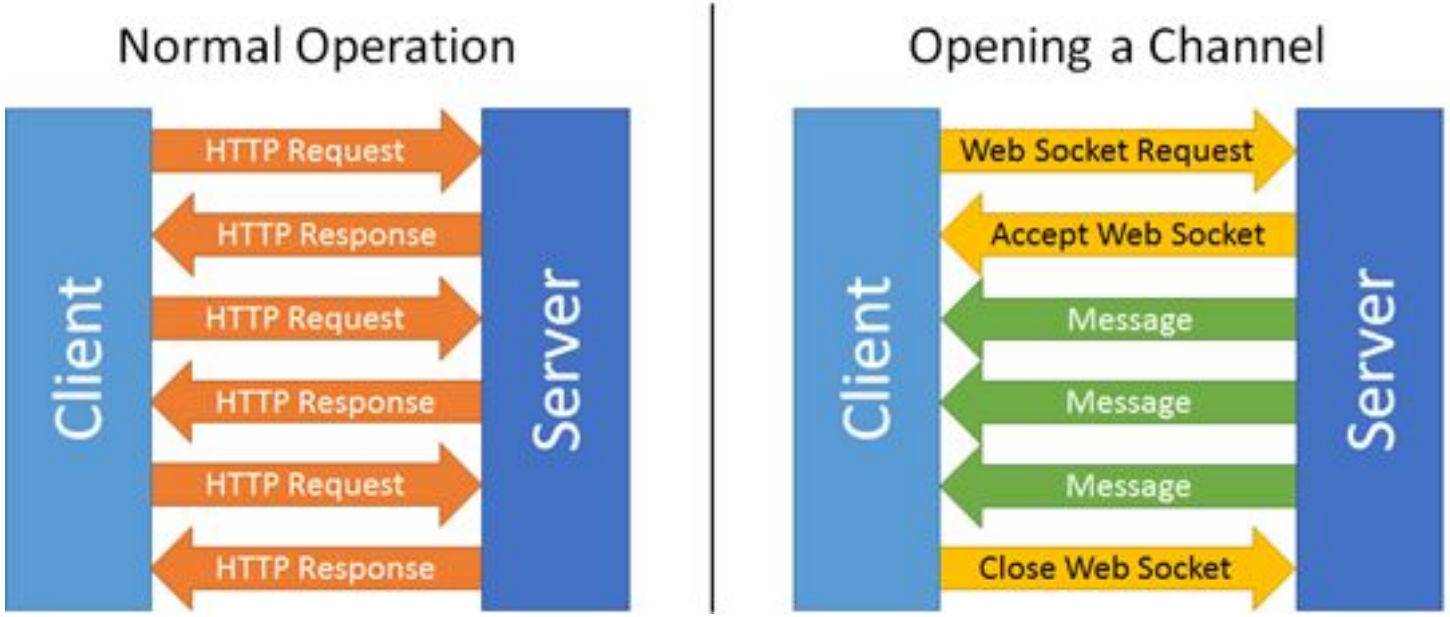
WebSocket

- **Disadvantage of WebSocket ?**
  - Still rely on TCP on L4 layer (Similar HTTP/2)
  - Many year belong to unstandard until 2011 (RFC 6455). So many library on any language with websocket will be multiple standard and let's developer survive among of them
  - Alway-on connection open on server that consume more and more resource
  - Not support retransmission procedure. This need handle by application itself or 3rd party library
  - Not well-known protocol than http. Many security device and load balance have problem with this ws: or wss:
  - Have significant conflict with HTTP/2, HTTP/3 server and equipment
  - Hard to handle this communication on unreliable/poor network, Device will shown down !!!

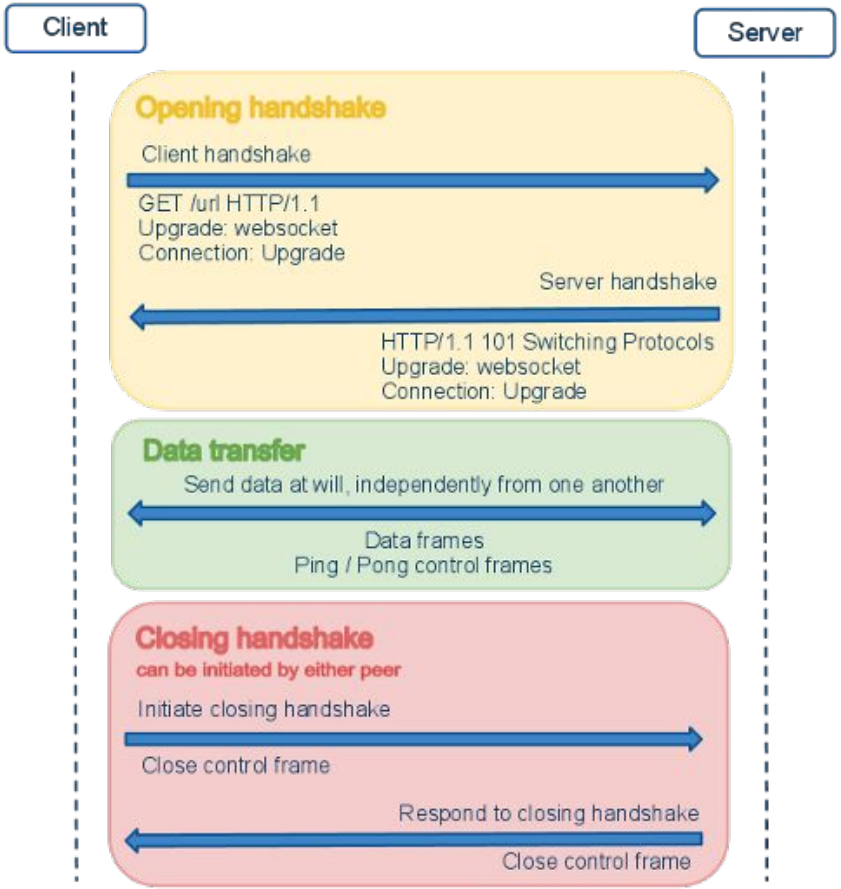
WebSocket

- **Is HTTP/2 , HTTP/3 will replace WebSocket ?**
  - HTTP/2 have push method (Server ⇒ Client).
  - HTTP/2 Compression header is very effective (HPAK) and reduce tcp pain point by HTTP/3 (QUIC)
  - HTTP/2 also support retransmission that make connection survive on poor network
  - In real world situation. Server can use push on HTTP/2 for process server send event (SSE) for push message to client
  - So HTTP will replace WebSocket ?
  - Answer is not really.
    - WebSocket will fully bi-directional, HTTP also half + server push
    - Some application still use WebSocket such as: firebase connection

WebSocket


Normal Operation
Client
Server
HTTP Request
HTTP Response
HTTP Request
HTTP Response
HTTP Request
HTTP Response
Opening a Channel
Client
Server
Web Socket Request
Accept Web Socket
Message
Message
Message
Close Web Socket

WebSocket


Client
Server
Opening handshake
Client handshake
GET /url HTTP/1.1
Upgrade: websocket
Connection: Upgrade
Server handshake
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Data transfer
Send data at will, independently from one another
Data frames
Ping / Pong control frames
Closing handshake
can be initiated by either peer
Initiate closing handshake
Close control frame
Respond to closing handshake
Close control frame

WebSocket

```
ubuntu@ip-10-21-2-22:~$ curl -i -N -H "Connection: Upgrade" -H "Upgrade: websocket" -H "Host: echo.websocket.org" -H "Origin: http://www.websocket.org" http://echo.websocket.org -v --http1.1
* Rebuilt URL to: http://echo.websocket.org/
*   Trying 174.129.224.73...
* TCP_NODELAY set
* Connected to echo.websocket.org (174.129.224.73) port 80 (#0)
> GET / HTTP/1.1
> Host: echo.websocket.org
> User-Agent: curl/7.58.0
> Accept: */*
> Connection: Upgrade
> Upgrade: websocket
> Origin: http://www.websocket.org
>
< HTTP/1.1 101 Web Socket Protocol Handshake
HTTP/1.1 101 Web Socket Protocol Handshake
< Access-Control-Allow-Credentials: true
Access-Control-Allow-Credentials: true
< Access-Control-Allow-Headers: content-type
Access-Control-Allow-Headers: content-type
< Access-Control-Allow-Headers: authorization
Access-Control-Allow-Headers: authorization
< Access-Control-Allow-Headers: x-websocket-extensions
Access-Control-Allow-Headers: x-websocket-extensions
< Access-Control-Allow-Headers: x-websocket-version
Access-Control-Allow-Headers: x-websocket-version
< Access-Control-Allow-Headers: x-websocket-protocol
Access-Control-Allow-Headers: x-websocket-protocol
< Access-Control-Allow-Origin: http://www.websocket.org
Access-Control-Allow-Origin: http://www.websocket.org
< Connection: Upgrade
Connection: Upgrade
< Date: Sun, 27 Dec 2020 08:18:43 GMT
Date: Sun, 27 Dec 2020 08:18:43 GMT
< Server: Kaazing Gateway
Server: Kaazing Gateway
< Upgrade: WebSocket
Upgrade: WebSocket
< WebSocket-Location: ws://echo.websocket.org/
WebSocket-Location: ws://echo.websocket.org/
< WebSocket-Origin: http://www.websocket.org
WebSocket-Origin: http://www.websocket.org

<
```

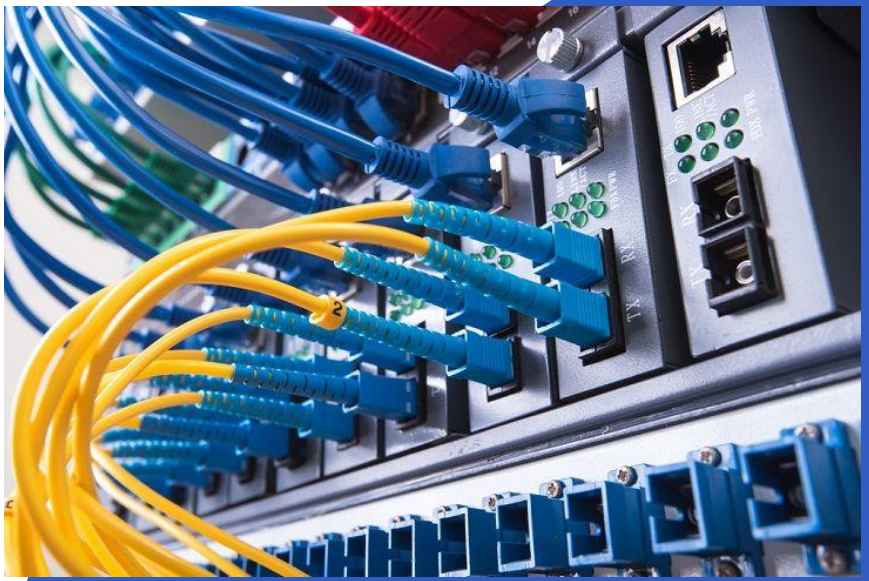
Workshop 2.4 WebSocket

```
ubuntu@ip-10-21-2-22:~$ curl -i -N -H "Connection: Upgrade" -H "Upgrade: websocket" -H "Host: echo.websocket.org" -H "Origin: http://www.websocket.org" http://echo.websocket.org -v --http1.1
* Rebuilt URL to: http://echo.websocket.org/
*   Trying 174.129.224.73...
* TCP_NODELAY set
* Connected to echo.websocket.org (174.129.224.73) port 80 (#0)
> GET / HTTP/1.1
> Host: echo.websocket.org
> User-Agent: curl/7.58.0
> Accept: */*
> Connection: Upgrade
> Upgrade: websocket
> Origin: http://www.websocket.org
>
< HTTP/1.1 101 Web Socket Protocol Handshake
HTTP/1.1 101 Web Socket Protocol Handshake
< Access-Control-Allow-Credentials: true
Access-Control-Allow-Credentials: true
< Access-Control-Allow-Headers: content-type
Access-Control-Allow-Headers: content-type
< Access-Control-Allow-Headers: authorization
Access-Control-Allow-Headers: authorization
< Access-Control-Allow-Headers: x-websocket-extensions
Access-Control-Allow-Headers: x-websocket-extensions
< Access-Control-Allow-Headers: x-websocket-version
Access-Control-Allow-Headers: x-websocket-version
< Access-Control-Allow-Headers: x-websocket-protocol
Access-Control-Allow-Headers: x-websocket-protocol
< Access-Control-Allow-Origin: http://www.websocket.org
Access-Control-Allow-Origin: http://www.websocket.org
< Connection: Upgrade
Connection: Upgrade
< Date: Sun, 27 Dec 2020 08:20:40 GMT
Date: Sun, 27 Dec 2020 08:20:40 GMT
< Server: Kaazing Gateway
Server: Kaazing Gateway
< Upgrade: WebSocket
Upgrade: WebSocket
< WebSocket-Location: ws://echo.websocket.org/
WebSocket-Location: ws://echo.websocket.org/
< WebSocket-Origin: http://www.websocket.org
WebSocket-Origin: http://www.websocket.org

<
```

Workshop 2.3 WebSocket

```
[ubuntu@ip-10-21-2-22:~$ sudo tcpdump -enni eth0 -vvvs 1024 port 80
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 1024 bytes
08:20:52.021911 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 64, id 33588, offset 0, flags [DF], proto TCP (6), length 60)
    10.21.2.22.38646 > 174.129.224.73.80: Flags [S], cksum 0x9b24 (incorrect -> 0xe9db), seq 1472638641, win 62727, options [mss 8961,sackOK,TS val 1943707456 ecr 0,nop,wscale 7], length 0
08:20:52.267262 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 44, id 0, offset 0, flags [DF], proto TCP (6), length 60)
    174.129.224.73.80 > 10.21.2.22.38646: Flags [S.], cksum 0x3022 (correct), seq 3438116691, ack 1472638642, win 26847, options [mss 1460,sackOK,TS val 77795898 ecr 1943707456,nop,wscale 7], length 0
08:20:52.267301 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 66: (tos 0x0, ttl 64, id 33589, offset 0, flags [DF], proto TCP (6), length 52)
    10.21.2.22.38646 > 174.129.224.73.80: Flags [.], cksum 0x9b1c (incorrect -> 0xc4ed), seq 1, ack 1, win 491, options [nop,nop,TS val 1943707701 ecr 77795898], length 0
08:20:52.267922 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 223: (tos 0x0, ttl 64, id 33590, offset 0, flags [DF], proto TCP (6), length 209)
    10.21.2.22.38646 > 174.129.224.73.80: Flags [P.], cksum 0x9bb9 (incorrect -> 0x83f2), seq 1:158, ack 1, win 491, options [nop,nop,TS val 1943707702 ecr 77795898], length 157: HTTP, length: 157
        GET / HTTP/1.1
        Host: echo.websocket.org
        User-Agent: curl/7.58.0
        Accept: */*
        Connection: Upgrade
        Upgrade: websocket
        Origin: http://www.websocket.org

08:20:52.513329 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 66: (tos 0x0, ttl 44, id 27939, offset 0, flags [DF], proto TCP (6), length 52)
    174.129.224.73.80 > 10.21.2.22.38646: Flags [.], cksum 0xc522 (correct), seq 1, ack 158, win 219, options [nop,nop,TS val 77795959 ecr 1943707702], length 0
08:20:52.517094 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 646: (tos 0x0, ttl 44, id 27940, offset 0, flags [DF], proto TCP (6), length 632)
    174.129.224.73.80 > 10.21.2.22.38646: Flags [P.], cksum 0x76d1 (correct), seq 1:581, ack 158, win 219, options [nop,nop,TS val 77795960 ecr 1943707702], length 580: HTTP, length: 580
        HTTP/1.1 101 Web Socket Protocol Handshake
        Access-Control-Allow-Credentials: true
        Access-Control-Allow-Headers: content-type
        Access-Control-Allow-Headers: authorization
        Access-Control-Allow-Headers: x-websocket-extensions
        Access-Control-Allow-Headers: x-websocket-version
        Access-Control-Allow-Headers: x-websocket-protocol
        Access-Control-Allow-Origin: http://www.websocket.org
        Connection: Upgrade
        Date: Sun, 27 Dec 2020 08:20:40 GMT
        Server: Kaazing Gateway
        Upgrade: WebSocket
        WebSocket-Location: ws://echo.websocket.org/
        WebSocket-Origin: http://www.websocket.org

08:20:52.517117 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 66: (tos 0x0, ttl 64, id 33591, offset 0, flags [DF], proto TCP (6), length 52)
    10.21.2.22.38646 > 174.129.224.73.80: Flags [.], cksum 0x9b1c (incorrect -> 0xc0d8), seq 158, ack 581, win 487, options [nop,nop,TS val 1943707951 ecr 77795960], length 0
```

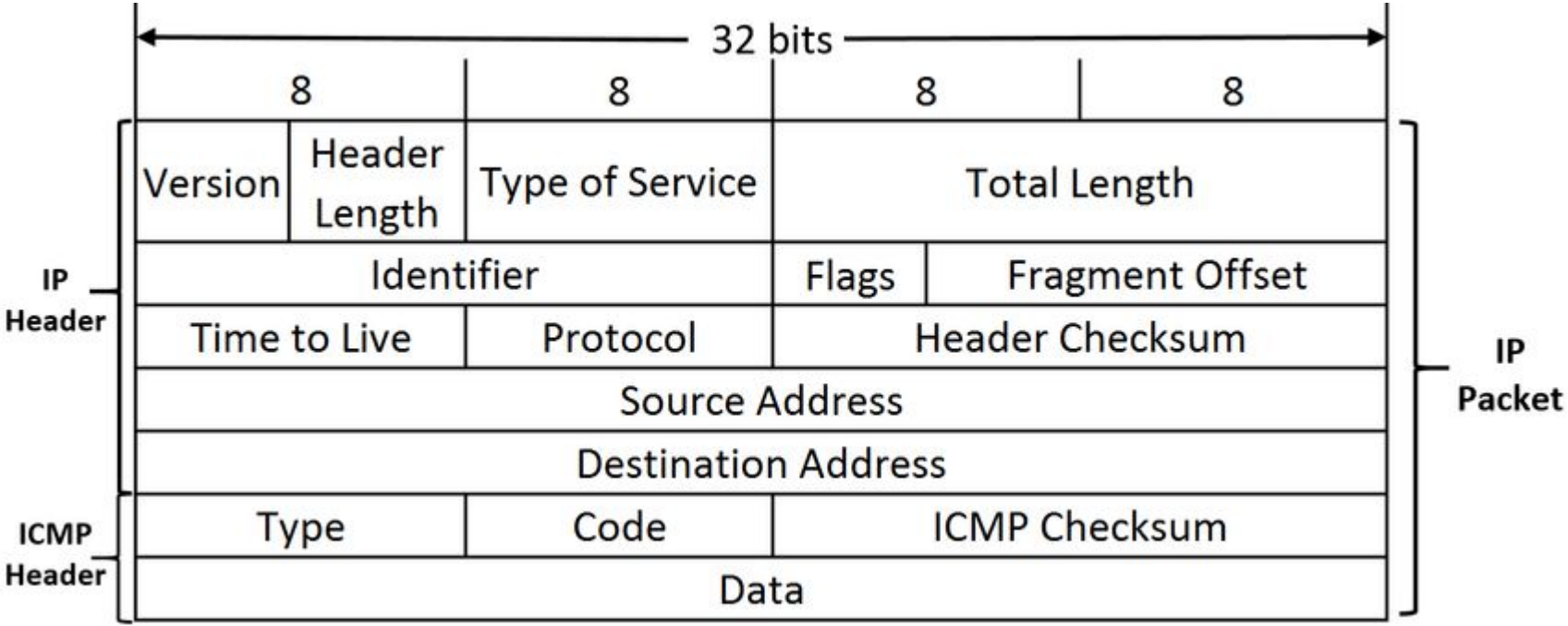
Advance Network Operation



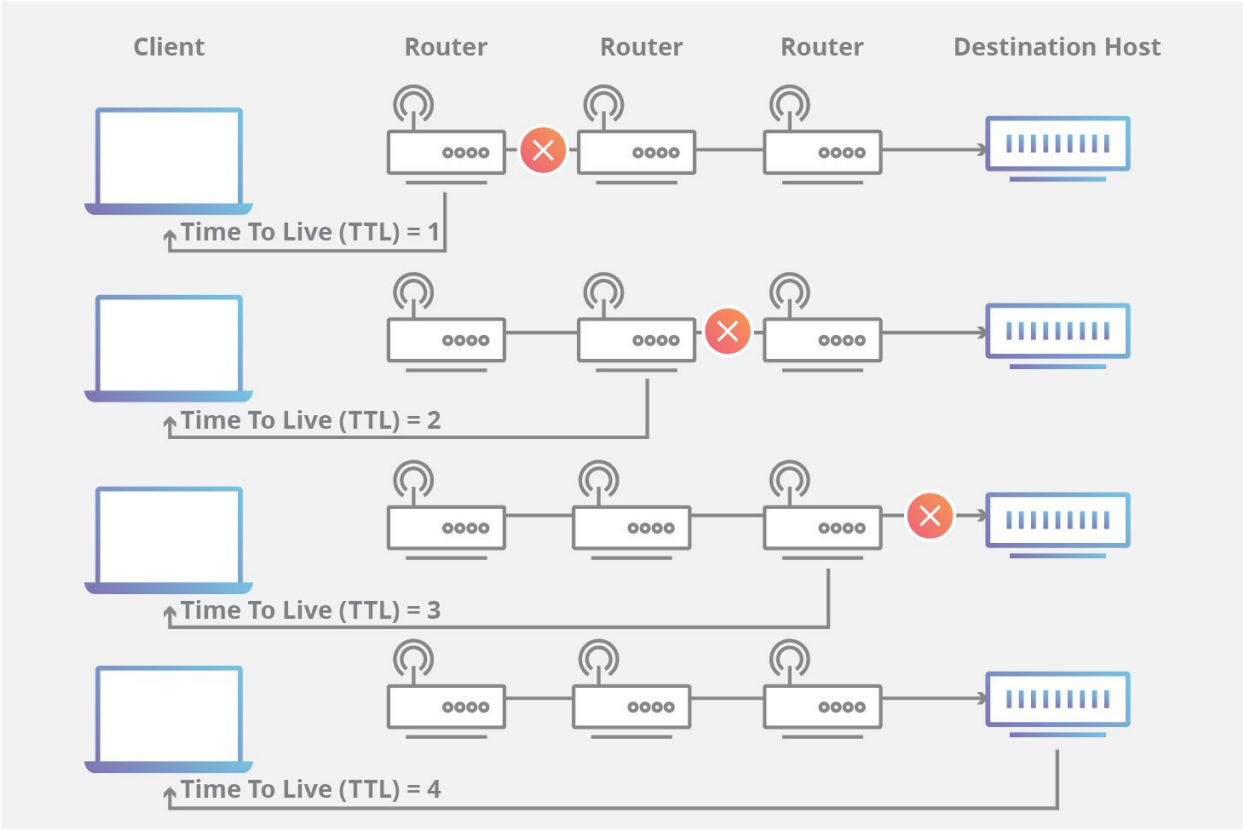
ICMP Operation

- **ICMP (Internet Control Message Protocol)**
  - General use for diagnostics , control purpose, response health, check available for network and security device
  - ICMP was classified on L3 and not belong with TCP/UDP
  - On real world it also integrate as part of ip packet.
  - Network and security device will process icmp packet as special purpose and separate from general ip packet
  - Traceroute for operate with ICMP for diagnostics network
  - ICMP also have timestamp for check round-trip time between sender and receiver
  - ICMP will consist with 4 elements
    - Type
    - Code
    - IP CheckSum
    - Data (4 byte depend of Code)

ICMP/TraceRoute Operation



ICMP/TraceRoute Operation



ICMP Operation

- ICMP type is define for control message
- ICMP code is additional context of message
- Example well-known type and code
  - Type: 0, Code: 0, Description: Echo Replay (Ping)
  - Type: 3, Code: 0, Description: Destination network unreachable
  - Type: 3, Code: 1, Description: Destination host unreachable
  - Type: 3, Code: 2, Description: Destination protocol unreachable
  - Type: 3, Code: 3, Description: Destination port unreachable
  - Type: 8, Code: 0, Description: Echo Request (Ping)
  - Type: 11, Code: 0, Description: TTL expired in transit (Traceroute)
  - Type: 11, Code: 1, Description: Fragment reassembly timeout exceed (Traceroute)

ICMP Operation

Type	Code	Status	Description
0 – Echo Reply[3]:14	0		Echo reply (used to ping)
1 and 2		unassigned	*Reserved*
3 – Destination Unreachable[3]:4	0		Destination network unreachable
	1		Destination host unreachable
	2		Destination protocol unreachable
	3		Destination port unreachable
	4		Fragmentation required, and DF flag set
	5		Source route failed
	6		Destination network unknown
	7		Destination host unknown
	8		Source host isolated
	9		Network administratively prohibited
	10		Host administratively prohibited
	11		Network unreachable for TOS
	12		Host unreachable for TOS
	13		Communication administratively prohibited
	14		Host Precedence Violation
	15		Precedence cutoff in effect
4 – Source Quench	0	deprecated	Source quench (congestion control)
5 – Redirect Message	0		Redirect Datagram for the Network
	1		Redirect Datagram for the Host
	2		Redirect Datagram for the TOS & network
	3		Redirect Datagram for the TOS & host
6		deprecated	Alternate Host Address
7		unassigned	*Reserved*
8 – Echo Request	0		Echo request (used to ping)
9 – Router Advertisement	0		Router Advertisement
10 – Router Solicitation	0		Router discovery/selection/solicitation
11 – Time Exceeded[3]:6	0		TTL expired in transit
	1		Fragment reassembly time exceeded

ICMP Operation

```
praparns-MacBook-Pro:AWS_Outpost praparn$ nslookup www.google.com
Server:         1.1.1.1
Address:        1.1.1.1#53

Non-authoritative answer:
Name:   www.google.com
Address: 172.217.166.132
```

```
praparns-MacBook-Pro:~ praparn$ traceroute -w 1 -m 100 172.217.166.132
traceroute to 172.217.166.132 (172.217.166.132), 100 hops max, 52 byte packets
 1  192.168.1.1 (192.168.1.1)  3.897 ms  2.034 ms  2.230 ms
 2  10.169.13.130 (10.169.13.130)  6.289 ms
    10.169.13.134 (10.169.13.134)  5.518 ms
    10.169.13.138 (10.169.13.138)  3.693 ms
 3  10.169.73.1 (10.169.73.1)  3.587 ms
    10.169.13.149 (10.169.13.149)  4.102 ms
    10.169.13.137 (10.169.13.137)  3.250 ms
 4  171-102-253-25.static.asianet.co.th (171.102.253.25)  9.005 ms
    171-102-248-117.static.asianet.co.th (171.102.248.117)  4.655 ms
    171-102-253-25.static.asianet.co.th (171.102.253.25)  6.788 ms
 5  171-102-249-112.static.asianet.co.th (171.102.249.112)  4.019 ms
    171-102-253-40.static.asianet.co.th (171.102.253.40)  6.466 ms
    171-102-249-112.static.asianet.co.th (171.102.249.112)  4.812 ms
 6  171-102-254-65.static.asianet.co.th (171.102.254.65)  6.297 ms
    171-102-250-1.static.asianet.co.th (171.102.250.1)  3.874 ms
    171-102-254-65.static.asianet.co.th (171.102.254.65)  5.165 ms
 7  171-102-254-232.static.asianet.co.th (171.102.254.232)  5.309 ms  6.469 ms  6.191 ms
 8  171-102-250-156.static.asianet.co.th (171.102.250.156)  12.387 ms  5.195 ms  5.321 ms
 9  tig-net25-204.trueintergateway.com (122.144.25.204)  4.793 ms
    tig-net25-210.trueintergateway.com (122.144.25.210)  4.755 ms  4.977 ms
10  tig-net246-5.trueintergateway.com (113.21.246.5)  23.242 ms
    tig-net246-49.trueintergateway.com (113.21.246.49)  22.132 ms  24.419 ms
11  72.14.197.30 (72.14.197.30)  22.633 ms  23.234 ms  22.284 ms
12  108.170.249.241 (108.170.249.241)  23.866 ms  24.286 ms  24.130 ms
13  72.14.234.85 (72.14.234.85)  23.798 ms  23.862 ms
    kul09s13-in-f4.1e100.net (172.217.166.132)  22.620 ms
praparns-MacBook-Pro:~ praparn$
```

Technical details

IP address	**172.217.166.132**
Hostname	kul09s13-in-f4.1e100.net
Type	Public
CIDR	172.217.166.132/24

Location of IP address 172.217.166.132

Lookup information about the location associated with the IP address 172.217.166.132.

City	Mountain View *(20% confidence)*
Metrocode	807 (California, San Francisco-Oakland-San Jose CA)
Subdivision	California (CA) *(60% confidence)*
Country	United States (US) *(99% confidence)*
Postalcode	94043 *(10% confidence)*
Continent	North America (NA)
Time zone	America/Los_Angeles

ASN and ISP for IP address 172.217.166.132

General traits like organisation, autonomous system number (ASN) and ISP associated with the IP address 172.217.166.132.

ISP	Google
Organization	Google
User type	business
Autonomous system number (ASN)	15169
Autonomous system organization	Google LLC
Anonymous proxy?	No
Satellite provider?	No

Registered and represented country

Details about the country in which the ISP has registered the IP address 172.217.166.132. Furthermore, if available, the represented country. For instance, the country represented by an overseas military base or embassy.

Registered country	United States (US)
Represented country	Not Provided

ICMP Operation

```
praparns-MacBook-Pro:~ praparn$ traceroute -w 1 -m 100 172.217.166.132 1500
traceroute to 172.217.166.132 (172.217.166.132), 100 hops max, 1500 byte packets
 1  192.168.1.1 (192.168.1.1)  3.061 ms  1.568 ms  1.498 ms
 2  10.169.13.158 (10.169.13.158)  22.528 ms
    10.169.13.166 (10.169.13.166)  3.684 ms
    10.169.13.134 (10.169.13.134)  3.951 ms
 3  10.169.13.141 (10.169.13.141)  3.368 ms  3.859 ms  3.504 ms
 4  171-102-253-25.static.asianet.co.th (171.102.253.25)  3.933 ms  4.007 ms  4.759 ms
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  tig-net25-206.trueintergateway.com (122.144.25.206)  6.578 ms
    tig-net25-204.trueintergateway.com (122.144.25.204)  6.888 ms  6.706 ms
10  tig-net246-49.trueintergateway.com (113.21.246.49)  23.729 ms  23.430 ms  23.328 ms
11  72.14.197.30 (72.14.197.30)  23.353 ms  24.190 ms  39.682 ms
12  * * *
13  108.170.225.238 (108.170.225.238)  37.083 ms  26.275 ms
    108.170.230.128 (108.170.230.128)  29.673 ms
14  108.170.250.10 (108.170.250.10)  24.842 ms
    72.14.234.89 (72.14.234.89)  24.960 ms
    72.14.234.85 (72.14.234.85)  23.990 ms
15  108.170.249.241 (108.170.249.241)  25.324 ms  24.643 ms
    kul09s13-in-f4.1e100.net (172.217.166.132)  27.619 ms
praparns-MacBook-Pro:~ praparn$
```

```
praparns-MacBook-Pro:~ praparn$ traceroute -w 1 -m 100 172.217.166.132 1492
traceroute to 172.217.166.132 (172.217.166.132), 100 hops max, 1492 byte packets
 1  192.168.1.1 (192.168.1.1)  1.879 ms  2.237 ms  1.454 ms
 2  10.169.13.154 (10.169.13.154)  5.039 ms
    10.169.13.158 (10.169.13.158)  4.530 ms
    10.169.13.162 (10.169.13.162)  6.104 ms
 3  10.169.73.9 (10.169.73.9)  3.911 ms
    10.169.13.161 (10.169.13.161)  4.445 ms
    10.169.13.165 (10.169.13.165)  4.071 ms
 4  171-102-248-117.static.asianet.co.th (171.102.248.117)  4.739 ms
    171-102-248-145.static.asianet.co.th (171.102.248.145)  5.515 ms
    171-102-248-117.static.asianet.co.th (171.102.248.117)  4.687 ms
 5  171-102-249-136.static.asianet.co.th (171.102.249.136)  4.370 ms
    171-102-249-160.static.asianet.co.th (171.102.249.160)  5.600 ms
    171-102-248-88.static.asianet.co.th (171.102.248.88)  5.430 ms
 6  171-102-250-1.static.asianet.co.th (171.102.250.1)  5.443 ms  5.287 ms  4.393 ms
 7  171-102-254-232.static.asianet.co.th (171.102.254.232)  5.137 ms  4.551 ms  4.654 ms
 8  171-102-250-156.static.asianet.co.th (171.102.250.156)  3.862 ms  5.358 ms  6.295 ms
 9  tig-net25-206.trueintergateway.com (122.144.25.206)  6.187 ms
    tig-net25-204.trueintergateway.com (122.144.25.204)  4.470 ms  5.110 ms
10  tig-net246-5.trueintergateway.com (113.21.246.5)  22.363 ms  23.772 ms
    tig-net246-49.trueintergateway.com (113.21.246.49)  23.041 ms
11  72.14.197.30 (72.14.197.30)  25.289 ms  29.082 ms  24.518 ms
12  108.170.249.225 (108.170.249.225)  22.662 ms  24.336 ms  22.479 ms
13  72.14.234.85 (72.14.234.85)  24.857 ms
    kul09s13-in-f4.1e100.net (172.217.166.132)  22.818 ms
    108.170.230.128 (108.170.230.128)  23.621 ms
praparns-MacBook-Pro:~ praparn$
```

ICMP Operation

Protocol overhead [edit]

PPPoE is used to connect a PC or a router to a modem via an Ethernet link and it can also be used in Internet access over DSL on a telephone line in the *PPPoE over ATM* (PPPoEoA) over ADSL protocol stack. PPPoE over ATM has the highest overhead of the popular DSL delivery methods, when compared with for example PPPoA (RFC 2364).[15][16][17][18]

Use with DSL – PPPoE over ATM (PPPoEoA) [edit]

The amount of overhead added by PPPoEoA on a DSL link depends on the packet size because of (i) the absorbing effect of ATM cell-padding (discussed below), which completely cancels out additional overhead of PPPoEoA in some cases, (ii) PPPoEoA + AAL5 overhead which can cause an entire additional 53-byte ATM cell to be required, and (iii) in the case of IP packets, PPPoE overhead added to packets that are near maximum length (*'MRU'*) may cause IP fragmentation, which also involves the first two considerations for both of the resulting IP fragments.[19] However ignoring ATM and IP fragmentation for the moment, the protocol header overheads for **ATM payload** due to choosing PPP + PPPoEoA can be as high as **44 bytes** = 2 bytes (for PPP) + 6 (for PPPoE) + 18 (Ethernet MAC, variable) + 10 (RFC 2684 LLC, variable) + 8 (AAL5 CPCS).[15] This overhead is that obtained when using the LLC header option described in RFC 2684 for PPPoEoA.[17][18]

Compare this with a vastly more header-efficient protocol, PPP + PPPoA RFC 2364 VC-MUX over ATM+DSL, which has a mere 10-byte overhead within the ATM payload. (In fact, just simply 10 bytes = 2 bytes for PPP + zero for RFC 2364 + 8 (AAL5 CPCS).)[16][18]

This figure of 44 bytes AAL5 payload overhead can be reduced in two ways: (i) by choosing the RFC 2684 option of discarding the 4-byte Ethernet MAC FCS, which reduces the figure of 18 bytes above to 14, and (ii) by using the RFC 2684 VC-MUX option, whose overhead contribution is a mere 2 bytes compared with the 10 byte overhead of the LLC alternative. It turns out that this overhead reduction can be a valuable efficiency improvement. Using VC-MUX instead of LLC, the ATM payload overhead is either 32 bytes (without Ethernet FCS) or 36 bytes (with FCS).[15][17]

ATM AAL5 requires that an 8-byte-long 'CPCS' trailer must always be present at the very end of the final cell ('right justified') of the run of ATM cells that make up the AAL5 payload packet. In the LLC case, the total ATM payload overhead is 2 + 6 + 18 + 10 + 8 = 44 bytes if the Ethernet MAC FCS is present, or 2 + 6 + 14 + 10 + 8 = 40 bytes with no FCS. In the more efficient VC-MUX case the ATM payload overhead is 2 + 6 + 18 + 2 + 8 = 36 bytes (with FCS), or 2 + 6 + 14 + 2 + 8 = **32 bytes** (no FCS).

However, the true overhead in terms of the total amount of ATM payload data sent is not simply a fixed additional value – it can *only be either zero or 48 bytes* (leaving aside scenario (iii) mentioned earlier, IP fragmentation). This is because ATM cells are fixed length with a payload capacity of 48 bytes, and adding a greater extra amount of AAL5 payload due to additional headers may require one more whole ATM cell to be sent containing the excess. The last one or two ATM cells contain padding bytes as required to ensure that each cell's payload is 48 bytes long.[15][17]

An example: In the case of a 1500-byte IP packet sent over AAL5/ATM with PPPoEoA and RFC2684-LLC, neglecting final cell padding for the moment, one starts with 1500 + 2 + 6 + 18 + 10 + 8 (AAL5 CPCS trailer) = 1544 bytes if the ethernet FCS is present, or else + 2 + 6 + 14 + 10 + 8 = 40 bytes with no FCS. To send 1544 bytes over ATM requires 33 48-byte ATM cells, since the available payload capacity of 32 cells × 48 bytes per cell = 1536 bytes is not quite enough. Compare this to the case of PPP + PPPoA which at 1500 + 2 (PPP) + 0 (PPPoA: RFC 2364 VC-MUX) + 8 (CPCS trailer) = 1510 bytes fits in 32 cells. So the real cost of choosing PPPoEoA plus RFC2684-LLC for 1500-byte IP packets is one additional ATM cell per IP packet, a ratio of 33:32.[15][16][17] So for 1500 byte packets, PPPoEoA with LLC is ~3.125% slower than PPPoA or optimal choices of PPPoEoA header options.

For some packet lengths the true additional effective DSL overhead due to choosing PPPoEoA compared with PPPoA will be zero if the extra header overhead is not enough to need an additional ATM cell at that particular packet length. For example, a 1492 byte long packet sent with PPP + PPPoEoA using RFC2684-LLC plus FCS gives us a total ATM payload of 1492 + 44 = 1536 bytes = 32 cells exactly, and the overhead in this special case is no greater than if we were using the header-efficient PPPoA protocol, which would require 1492 + 2 + 0 + 8 = 1502 bytes ATM payload = 32 cells also.[15][17] The case where the packet length is 1492 represents the optimum efficiency for PPPoEoA with RFC2684-LLC in ratio terms, unless even longer packets are allowed.

Using PPPoEoA with the RFC2684 VC-MUX header option is always much more efficient than the LLC option, since the ATM overhead, as mentioned earlier, is only 32 or 36 bytes (depending on whether this is without or with the ethernet FCS option in PPPoEoA) so that a 1500 byte long packet including all overheads of PPP + PPPoEoA using VC-MUX equates to a total 1500 + 36 = 1536 bytes ATM payload if the FCS is present = 32 ATM cells exactly, thus saving an entire ATM cell.[15][17]

With short packets, the longer the header overheads the greater the likelihood of generating an additional ATM cell. A worst case might be sending 3 ATM cells instead of two because of a 44 byte header overhead compared with a 10 byte header overhead, so 50% more time taken to transmit the data. For example, a TCP ACK packet over IPv6 is 60 bytes long, and with overhead of 40 or 44 bytes for PPPoEoA + LLC this requires three 48 byte ATM cells' payloads. As a comparison, PPPoA with overheads of 10 bytes so 70 bytes total fits into two cells. So the extra cost of choosing PPPoE/LLC over PPPoA is 50% extra data sent. PPPoEoA + VC-MUX would be fine though: with 32 or 36 byte overhead, our IP packet still fits in two cells.

In all cases the most efficient option for ATM-based ADSL internet access is to choose PPPoA (RFC2364) VC-MUX. However, if PPPoEoA is required, then the best choice is always to use VC-MUX (as opposed to LLC) with no Ethernet FCS, giving an ATM payload overhead of **32 bytes** = 2 bytes (for PPP) + 6 (for PPPoE) + 14 (Ethernet MAC, no FCS) + 2 (RFC 2684 VC-MUX) + 8 (AAL5 CPCS trailer).

Unfortunately some DSL services require the use of wasteful LLC headers with PPPoE and do not allow the more efficient VC-MUX option. In that case, using a reduced packet length, such as enforcing a maximum MTU of 1492 regains efficiency with long packets even with LLC headers and, as mentioned earlier, in that case no extra wasteful ATM cell is generated.

Overhead on Ethernet [edit]

On an Ethernet LAN, the overhead for PPP + PPPoE is a fixed 2 + 6 = **8 bytes**, unless IP fragmentation is produced.

ICMP Operation

- **ICMP (Internet Control Message Protocol)**
  - General use for diagnostics , control purpose, response health, check available for network and security device
  - ICMP was classified on L3 and not belong with TCP/UDP
  - On real world it also integrate as part of ip packet.
  - Network and security device will process icmp packet as special purpose and separate from general ip packet
  - Traceroute for operate with ICMP for diagnostics network
  - ICMP also have timestamp for check round-trip time between sender and receiver
  - ICMP will consist with 4 elements
    - Type
    - Code
    - IP CheckSum
    - Data (4 byte depend of Code)

Workshop 2.4 ICMP and TraceRoute Operation

```
ubuntu@ip-10-21-2-22:~$ dig @8.8.8.8 www.google.com A

; <<>> DiG 9.11.3-1ubuntu1.13-Ubuntu <<>> @8.8.8.8 www.google.com A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22460
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.google.com.                        IN      A

;; ANSWER SECTION:
www.google.com.         209     IN      A       74.125.24.103
www.google.com.         209     IN      A       74.125.24.106
www.google.com.         209     IN      A       74.125.24.147
www.google.com.         209     IN      A       74.125.24.105
www.google.com.         209     IN      A       74.125.24.104
www.google.com.         209     IN      A       74.125.24.99

;; Query time: 2 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Sun Dec 27 08:40:51 UTC 2020
;; MSG SIZE  rcvd: 139

ubuntu@ip-10-21-2-22:~$ ping -c 10 -i 1 -W 1 74.125.24.103
PING 74.125.24.103 (74.125.24.103) 56(84) bytes of data.
64 bytes from 74.125.24.103: icmp_seq=1 ttl=99 time=2.38 ms
64 bytes from 74.125.24.103: icmp_seq=2 ttl=99 time=2.42 ms
64 bytes from 74.125.24.103: icmp_seq=3 ttl=99 time=2.43 ms
64 bytes from 74.125.24.103: icmp_seq=4 ttl=99 time=2.51 ms
64 bytes from 74.125.24.103: icmp_seq=5 ttl=99 time=2.42 ms
64 bytes from 74.125.24.103: icmp_seq=6 ttl=99 time=2.36 ms
64 bytes from 74.125.24.103: icmp_seq=7 ttl=99 time=2.43 ms
64 bytes from 74.125.24.103: icmp_seq=8 ttl=99 time=2.41 ms
64 bytes from 74.125.24.103: icmp_seq=9 ttl=99 time=2.38 ms
64 bytes from 74.125.24.103: icmp_seq=10 ttl=99 time=2.38 ms

--- 74.125.24.103 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9015ms
rtt min/avg/max/mdev = 2.360/2.415/2.519/0.071 ms
ubuntu@ip-10-21-2-22:~$
```

```
ubuntu@ip-10-21-2-22:~$ sudo tcpdump -enni eth0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
08:41:09.261040 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 1, length 64
08:41:09.263417 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 1, length 64
08:41:10.262739 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 2, length 64
08:41:10.265139 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 2, length 64
08:41:11.264487 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 3, length 64
08:41:11.266895 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 3, length 64
08:41:12.266210 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 4, length 64
08:41:12.268703 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 4, length 64
08:41:13.268024 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 5, length 64
08:41:13.270428 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 5, length 64
08:41:14.269812 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 6, length 64
08:41:14.272151 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 6, length 64
08:41:15.271474 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 7, length 64
08:41:15.273884 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 7, length 64
08:41:16.273210 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 8, length 64
08:41:16.275605 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 8, length 64
08:41:17.274926 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 9, length 64
08:41:17.277285 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 9, length 64
08:41:18.276598 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 10, length 64
08:41:18.278963 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 10, length 64
^C
20 packets captured
20 packets received by filter
0 packets dropped by kernel
```

```
ubuntu@ip-10-21-2-22:~$ traceroute -w 2 -m 100 -q 10 74.125.24.103
traceroute to 74.125.24.103 (74.125.24.103), 100 hops max
  1   175.41.128.185  3.457ms  4.266ms  25.805ms  3.921ms  1.131ms  1.130ms  1.809ms  8.443ms  8.099ms  8.144ms
  2   100.65.32.80  2.633ms  8.135ms  8.150ms  8.087ms  8.093ms  8.198ms  8.147ms  8.114ms  8.116ms  8.054ms
  3   100.66.16.122  3.495ms  8.090ms  8.138ms  2.962ms  8.164ms  8.092ms  8.202ms  8.183ms  8.143ms  8.577ms
  4   100.66.18.234  12.956ms  21.955ms  21.945ms  18.336ms  19.908ms  18.463ms  21.897ms  21.992ms  21.899ms  21.940ms
  5   100.66.7.73  16.504ms  23.264ms  21.488ms  22.147ms  21.931ms  21.934ms  11.814ms  21.780ms  21.983ms  21.945ms
  6   100.66.4.93  18.966ms  20.794ms  19.948ms  21.937ms  21.865ms  21.983ms  21.851ms  21.928ms  21.894ms  21.882ms
  7   100.65.10.129  0.444ms  0.382ms  0.407ms  0.383ms  0.377ms  0.389ms  0.374ms  0.392ms  0.925ms  0.424ms
  8   52.93.10.74  1.335ms  1.314ms  1.668ms  1.593ms  1.308ms  1.323ms  1.276ms  1.247ms  1.432ms  1.493ms
  9   52.93.8.72  1.815ms  *  29.540ms  *  *  1.656ms  *  *  *  *
 10   52.93.10.145  5.514ms  11.716ms  9.460ms  2.840ms  1.545ms  1.240ms  1.247ms  1.421ms  1.533ms  2.202ms
 11   99.82.177.15  3.417ms  3.426ms  3.231ms  3.352ms  3.372ms  3.369ms  3.377ms  3.383ms  3.414ms  3.343ms
 12   108.170.254.226  2.928ms  2.729ms  2.722ms  2.685ms  3.034ms  2.758ms  2.621ms  2.740ms  2.674ms  2.643ms
 13   66.249.95.248  2.907ms  2.867ms  2.909ms  3.045ms  3.075ms  2.994ms  2.901ms  2.911ms  2.816ms  2.899ms
 14   108.170.240.242  2.866ms  3.048ms  2.931ms  2.942ms  3.017ms  2.960ms  2.914ms  3.032ms  2.864ms  2.902ms
 15   216.239.35.148  4.710ms  5.888ms  *  2.569ms  2.427ms  2.643ms  *  2.552ms  *  2.713ms
 16   216.239.51.20  19.038ms  17.846ms  8.806ms  15.055ms  7.574ms  13.310ms  14.413ms  2.561ms  2.324ms  2.434ms
 17   *  *  *  *  *  *  *  *  *  *
 18   *  *  *  *  *  *  *  *  *  *
 19   *  *  *  *  *  *  *  *  *  *
 20   *  *  *  *  *  *  *  *  *  *
 21   *  *  *  *  *  *  *  *  *  *
 22   *  *  *  *  *  *  *  *  *  *
 23   *  *  *  *  *  *  *  *  *  *
 24   *  *  *  *  *  *  *  *  *  *
 25   *  *  *  *  *  *  *  *  *  *
 26   74.125.24.103  2.903ms  2.564ms  2.584ms  2.558ms  2.528ms  2.560ms  2.538ms  2.513ms  2.507ms  2.491ms
ubuntu@ip-10-21-2-22:~$
```

Linux IPTables Operation

- Since linux kernel 2.4 (2001+). The “NetFilter” was introduce for packet management infrastructure and “IPTables” is administrator tool for manage this
- IPTables will manage packet with “chain of rule”
- Rule on iptables will process by sequential on each chain of packet
- Every packet arrive/leaving to computer will reach at least on chain
- IPTables have 5 predefine chain (map with NetFilter hook event)
    - **PREROUTING:** Packet enter before routing decision
    - **INPUT:** Packet go to machine locally and will start open socket for service
    - **FORWARD:** Packet that route to other node will reach this chain
    - **OUTPUT:** Packet sent out from machine to destination
    - **POSTROUTING:** Packet was routed and event is before sent to network card

Linux IPTables Operation

- IPTables can configure response (jump to other chain) for each rule like action below
  - **ACCEPT:** Allow connection
  - **DROP:** Just drop connection with silent notify. This best option in case we don't want source recognized your action
  - **REJECT:** Denied connection and send result back to source
  - **LOG:** Log packet and continue
- IPTables also support state management (TCP state) for allow incoming and outgoing with same state for operate like below:
  - **NEW:** Allow new connection
  - **ESTABLISHED:** Allow the connection that was sync state (three way connection)
  - **RELATED:** Allow new connection that related to other state (such as FTP-Data, ICMP-Error etc)
  - **INVALID**: Cannot identified with some reason else

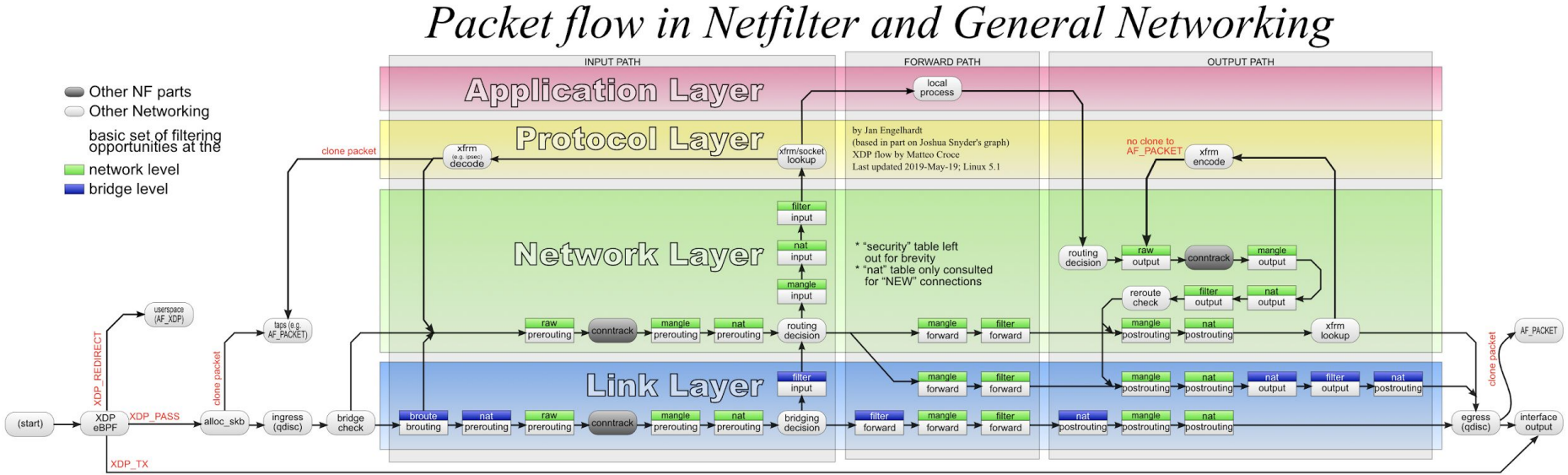
Linux IPTables Operation

- Other generic option on iptables
  - **-A:** Append new rule
  - **-L:** List current iptables
  - **-m state:** Use with --state for work with connection state
  - **--state:** define state for action (NEW, ESTABLISHED, RELATED, INVALID)
  - **--limit:** Limit action with number of time
  - **-P:** Persistent rule (Default)
  - **-p:** Protocol (tcp,udp etc)
  - **--sport:** Source port
  - **--dport:** Destination port
  - **-j:** jump to chain (ACCEPT, REJECT, LOG, DROP)
  - **--log-prefix:** add prefix to log
  - **--log-level:** log level (Normal is 7)
  - -i: interface card for apply

Linux IPTables Operation

- Other generic option on iptables
  - **-I:** Insert rule on between previous rule (Can specific rule no)
  - **-v:** Verbal display as output
  - **-s:** Identify source ip address
  - **-d:** Identify destination ip address
  - **-o:** Output interface

Linux IPTables Operation



Linux IPTables Operation

Standard Linux

```
ubuntu@ip-10-21-2-140:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ubuntu@ip-10-21-2-140:~$
```

Docker Linux

```
ubuntu@ip-10-21-2-114:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination
DOCKER-USER  all  --  anywhere             anywhere
DOCKER-ISOLATION-STAGE-1  all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ESTABLISHED
DOCKER     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ESTABLISHED
DOCKER     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ESTABLISHED
DOCKER     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain DOCKER (3 references)
target     prot opt source               destination

Chain DOCKER-ISOLATION-STAGE-1 (1 references)
target     prot opt source               destination
DOCKER-ISOLATION-STAGE-2  all  --  anywhere             anywhere
DOCKER-ISOLATION-STAGE-2  all  --  anywhere             anywhere
DOCKER-ISOLATION-STAGE-2  all  --  anywhere             anywhere
RETURN     all  --  anywhere             anywhere

Chain DOCKER-ISOLATION-STAGE-2 (3 references)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere
DROP       all  --  anywhere             anywhere
DROP       all  --  anywhere             anywhere
RETURN     all  --  anywhere             anywhere

Chain DOCKER-USER (1 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere
ubuntu@ip-10-21-2-114:~$
```

Example: IPTables

- Step1: (Server) Install NMAP for test operate:

Server

Client

```
ubuntu@ip-10-21-2-140:~$ sudo apt install -y nmap
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-aws-5.4-headers-5.4.0-1025 linux-aws-5.4-headers-5.4.0-1028 linux-aws-5.4-headers-5.4.0-1029
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libblas3 liblinear3 liblua5.3-0
Suggested packages:
  liblinear-tools liblinear-dev ndiff
The following NEW packages will be installed:
  libblas3 liblinear3 liblua5.3-0 nmap
0 upgraded, 4 newly installed, 0 to remove and 33 not upgraded.
Need to get 5467 kB of archives.
After this operation, 25.0 MB of additional disk space will be used.
Get:1 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu bionic/main amd64 libblas3 amd64 3.7.1-4ubuntu1 [140 kB]
Get:2 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu bionic/main amd64 liblinear3 amd64 2.1.0+dfsg-2 [39.3 kB]
Get:3 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 liblua5.3-0 amd64 5.3.3-1ubuntu0.18.04.1 [115 kB
]
Get:4 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu bionic/main amd64 nmap amd64 7.60-1ubuntu5 [5174 kB]
Fetched 5467 kB in 1s (9364 kB/s)
Selecting previously unselected package libblas3:amd64.
(Reading database ... 180820 files and directories currently installed.)
Preparing to unpack .../libblas3_3.7.1-4ubuntu1_amd64.deb ...
Unpacking libblas3:amd64 (3.7.1-4ubuntu1) ...
Selecting previously unselected package liblinear3:amd64.
Preparing to unpack .../liblinear3_2.1.0+dfsg-2_amd64.deb ...
Unpacking liblinear3:amd64 (2.1.0+dfsg-2) ...
Selecting previously unselected package liblua5.3-0:amd64.
Preparing to unpack .../liblua5.3-0_5.3.3-1ubuntu0.18.04.1_amd64.deb ...
Unpacking liblua5.3-0:amd64 (5.3.3-1ubuntu0.18.04.1) ...
Selecting previously unselected package nmap.
Preparing to unpack .../nmap_7.60-1ubuntu5_amd64.deb ...
Unpacking nmap (7.60-1ubuntu5) ...
Setting up libblas3:amd64 (3.7.1-4ubuntu1) ...
update-alternatives: using /usr/lib/x86_64-linux-gnu/blas/libblas.so.3 to provide /usr/lib/x86_64-linux-gnu/libblas.so.3 (libblas.so
.3-x86_64-linux-gnu) in auto mode
Setting up liblinear3:amd64 (2.1.0+dfsg-2) ...
Setting up liblua5.3-0:amd64 (5.3.3-1ubuntu0.18.04.1) ...
Setting up nmap (7.60-1ubuntu5) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for libc-bin (2.27-3ubuntu1.2) ...
ubuntu@ip-10-21-2-140:~$
```

Example: IPTables

- Step2: (Server) Apply standard iptables for block input as default (except 22), check public ip of server (recorded) and set hostname for

Server

```
ubuntu@ip-10-21-2-140:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ubuntu@ip-10-21-2-140:~$ curl ifconfig.co
13.229.61.229
ubuntu@ip-10-21-2-140:~$ sudo sed -i "s/127.0.0.1 localhost/127.0.0.1 localhost $HOSTNAME/g" /etc/hosts
ubuntu@ip-10-21-2-140:~$ sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
ubuntu@ip-10-21-2-140:~$ sudo iptables -P INPUT DROP
ubuntu@ip-10-21-2-140:~$ sudo iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ubuntu@ip-10-21-2-140:~$
```

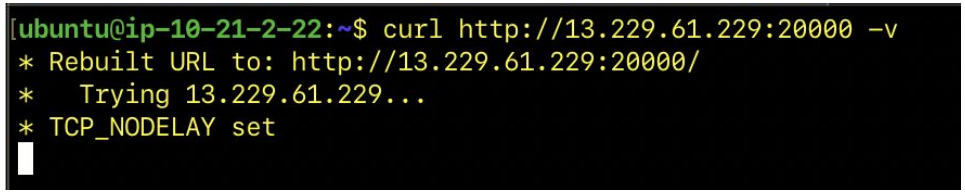
Example: IPTables

- Step3: (Server) Run ncat for listening port 20000 on server

Server

- Step4: (Client) Check public ip client (recorded) and curl to server with port 20000

Client



Example: IPTables

- Step5: (Server) ACCEPT iptables rule for allow connection from client (public ip address) to server with port 20000
  - INPUT: Accept connection (NEW,ESTABLISHED) to destination port 20000 with “I” for insert on first rule
  - OUTPUT: Accept connection (ESTABLISHED) for reply back to client with source port 20000 with “A” for append

Server

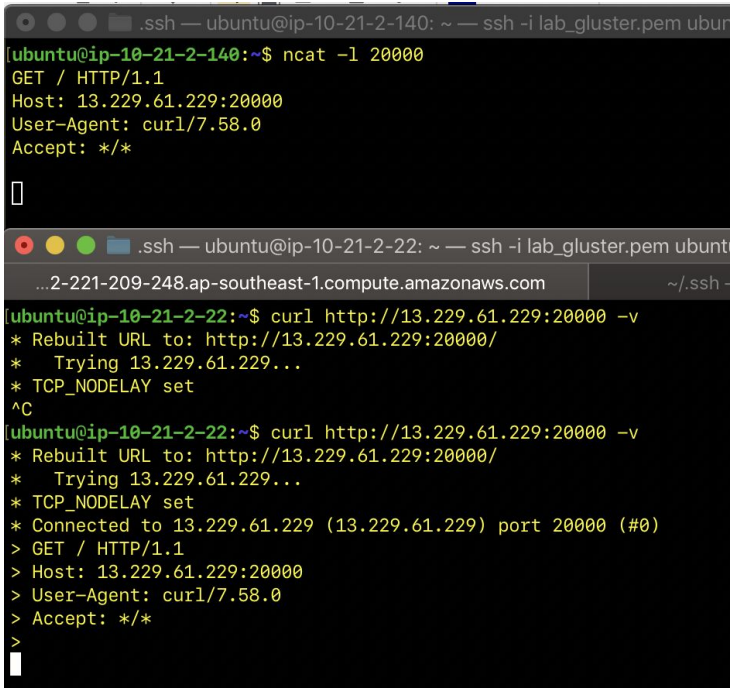
```
ubuntu@ip-10-21-2-140:~$ sudo iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ubuntu@ip-10-21-2-140:~$ sudo iptables -I INPUT 1 -p tcp --dport 20000 -m state --state NEW,ESTABLISHED -j ACCEPT
ubuntu@ip-10-21-2-140:~$ sudo iptables -A OUTPUT -p tcp --sport 20000 -m state --state ESTABLISHED -j ACCEPT
ubuntu@ip-10-21-2-140:~$ sudo iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:20000 state NEW,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp spt:20000 state ESTABLISHED
ubuntu@ip-10-21-2-140:~$
```

Example: IPTables

- Step6: (Server & Client) Run netcat for listening port 20000 on server and test with client again

Server

Client



Example: IPTables

- Step7: (Server) Remove iptables for rollback configuration

Server

```
ubuntu@ip-10-21-2-140:~$ sudo iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:20000 state NEW,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp spt:20000 state ESTABLISHED
ubuntu@ip-10-21-2-140:~$ sudo iptables -P INPUT ACCEPT
ubuntu@ip-10-21-2-140:~$ sudo iptables -D INPUT -p tcp --dport 22 -j ACCEPT
ubuntu@ip-10-21-2-140:~$ sudo iptables -D INPUT -p tcp --dport 20000 -m state --state NEW,ESTABLISHED -j ACCEPT
ubuntu@ip-10-21-2-140:~$ sudo iptables -D OUTPUT -p tcp --sport 20000 -m state --state ESTABLISHED -j ACCEPT
ubuntu@ip-10-21-2-140:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ubuntu@ip-10-21-2-140:~$
```

Workshop 2.5 IPTables Operation

```
ubuntu@ip-10-21-2-22:~$ dig @8.8.8.8 www.google.com A

; <<>> DiG 9.11.3-1ubuntu1.13-Ubuntu <<>> @8.8.8.8 www.google.com A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22460
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.google.com.                        IN      A

;; ANSWER SECTION:
www.google.com.         209     IN      A       74.125.24.103
www.google.com.         209     IN      A       74.125.24.106
www.google.com.         209     IN      A       74.125.24.147
www.google.com.         209     IN      A       74.125.24.105
www.google.com.         209     IN      A       74.125.24.104
www.google.com.         209     IN      A       74.125.24.99

;; Query time: 2 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Sun Dec 27 08:40:51 UTC 2020
;; MSG SIZE  rcvd: 139

ubuntu@ip-10-21-2-22:~$ ping -c 10 -i 1 -W 1 74.125.24.103
PING 74.125.24.103 (74.125.24.103) 56(84) bytes of data.
64 bytes from 74.125.24.103: icmp_seq=1 ttl=99 time=2.38 ms
64 bytes from 74.125.24.103: icmp_seq=2 ttl=99 time=2.42 ms
64 bytes from 74.125.24.103: icmp_seq=3 ttl=99 time=2.43 ms
64 bytes from 74.125.24.103: icmp_seq=4 ttl=99 time=2.51 ms
64 bytes from 74.125.24.103: icmp_seq=5 ttl=99 time=2.42 ms
64 bytes from 74.125.24.103: icmp_seq=6 ttl=99 time=2.36 ms
64 bytes from 74.125.24.103: icmp_seq=7 ttl=99 time=2.43 ms
64 bytes from 74.125.24.103: icmp_seq=8 ttl=99 time=2.41 ms
64 bytes from 74.125.24.103: icmp_seq=9 ttl=99 time=2.38 ms
64 bytes from 74.125.24.103: icmp_seq=10 ttl=99 time=2.38 ms

--- 74.125.24.103 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9015ms
rtt min/avg/max/mdev = 2.360/2.415/2.519/0.071 ms
ubuntu@ip-10-21-2-22:~$
```

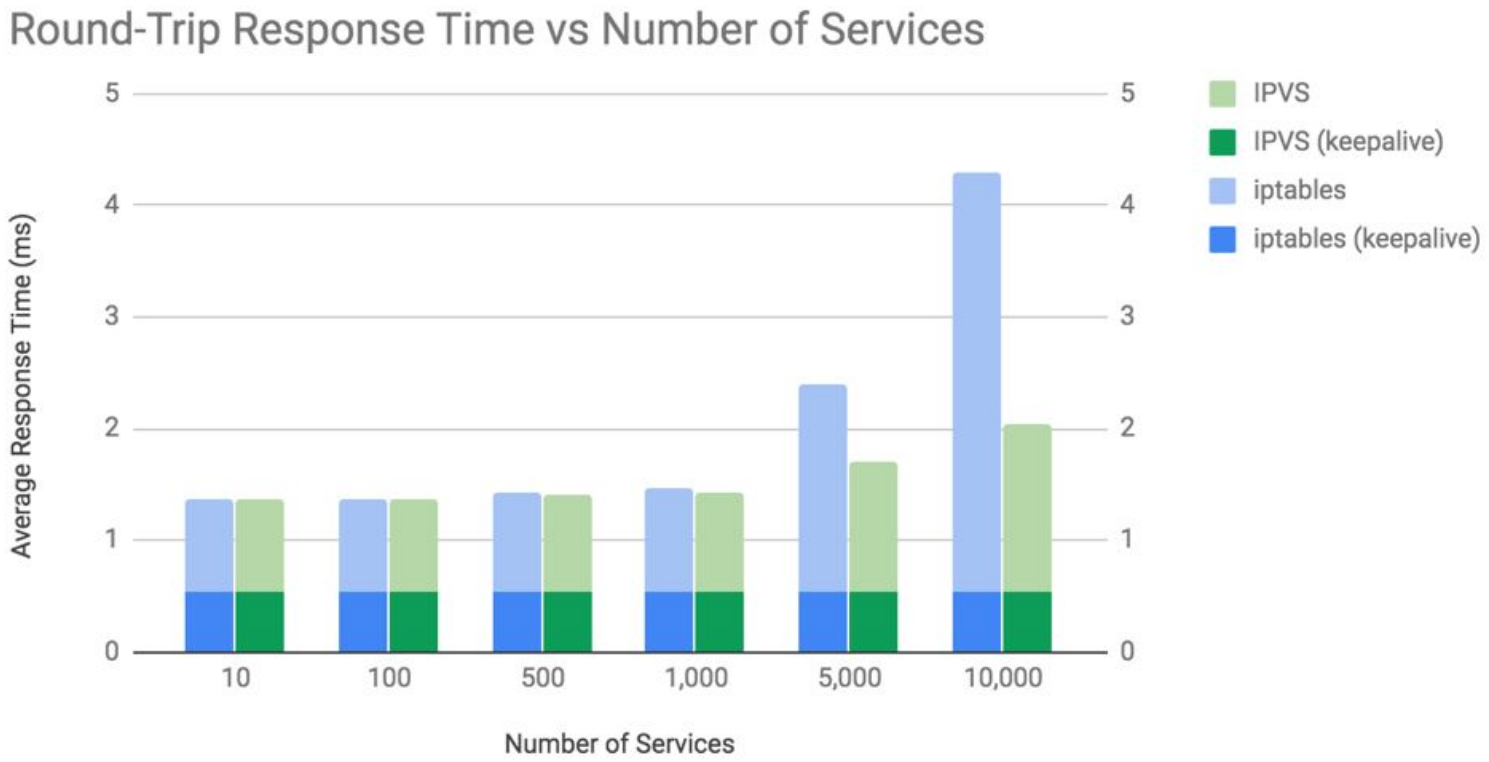
```
ubuntu@ip-10-21-2-22:~$ sudo tcpdump -enni eth0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
08:41:09.261040 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 1, length 64
08:41:09.263417 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 1, length 64
08:41:10.262739 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 2, length 64
08:41:10.265139 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 2, length 64
08:41:11.264487 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 3, length 64
08:41:11.266895 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 3, length 64
08:41:12.266210 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 4, length 64
08:41:12.268703 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 4, length 64
08:41:13.268024 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 5, length 64
08:41:13.270428 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 5, length 64
08:41:14.269812 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 6, length 64
08:41:14.272151 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 6, length 64
08:41:15.271474 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 7, length 64
08:41:15.273884 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 7, length 64
08:41:16.273210 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 8, length 64
08:41:16.275605 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 8, length 64
08:41:17.274926 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 9, length 64
08:41:17.277285 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 9, length 64
08:41:18.276598 06:36:ae:aa:e3:78 > 06:4d:1e:bf:fe:8e, ethertype IPv4 (0x0800), length 98: 10.21.2.22 > 74.125.24.103: ICMP echo request, id 2709, seq 10, length 64
08:41:18.278963 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 98: 74.125.24.103 > 10.21.2.22: ICMP echo reply, id 2709, seq 10, length 64
^C
20 packets captured
20 packets received by filter
0 packets dropped by kernel
```

```
ubuntu@ip-10-21-2-22:~$ sudo tcpdump -enni eth0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
09:00:02.241132 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 70: 52.93.10.133 > 10.21.2.22: ICMP time exceeded in-transit, length 36
09:00:02.245760 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 70: 52.93.10.133 > 10.21.2.22: ICMP time exceeded in-transit, length 36
09:00:02.247911 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 70: 52.93.10.133 > 10.21.2.22: ICMP time exceeded in-transit, length 36
09:00:02.249642 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 79: 74.125.118.250 > 10.21.2.22: ICMP time exceeded in-transit, length 45
09:00:02.252859 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 79: 74.125.118.250 > 10.21.2.22: ICMP time exceeded in-transit, length 45
09:00:02.255009 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 79: 74.125.118.250 > 10.21.2.22: ICMP time exceeded in-transit, length 45
09:00:17.273537 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 110: 108.170.254.225 > 10.21.2.22: ICMP time exceeded in-transit, length 76
09:00:17.276207 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 110: 108.170.254.225 > 10.21.2.22: ICMP time exceeded in-transit, length 76
09:00:17.278891 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 110: 108.170.254.225 > 10.21.2.22: ICMP time exceeded in-transit, length 76
09:00:17.281332 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 110: 74.125.242.34 > 10.21.2.22: ICMP time exceeded in-transit, length 76
09:00:17.283759 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 110: 74.125.242.34 > 10.21.2.22: ICMP time exceeded in-transit, length 76
09:00:17.286137 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 110: 74.125.242.34 > 10.21.2.22: ICMP time exceeded in-transit, length 76
09:00:17.288650 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 79: 66.249.94.59 > 10.21.2.22: ICMP time exceeded in-transit, length 45
09:00:17.291155 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 79: 66.249.94.59 > 10.21.2.22: ICMP time exceeded in-transit, length 45
09:00:17.293796 06:4d:1e:bf:fe:8e > 06:36:ae:aa:e3:78, ethertype IPv4 (0x0800), length 79: 66.249.94.59 > 10.21.2.22: ICMP time exceeded in-transit, length 45
```

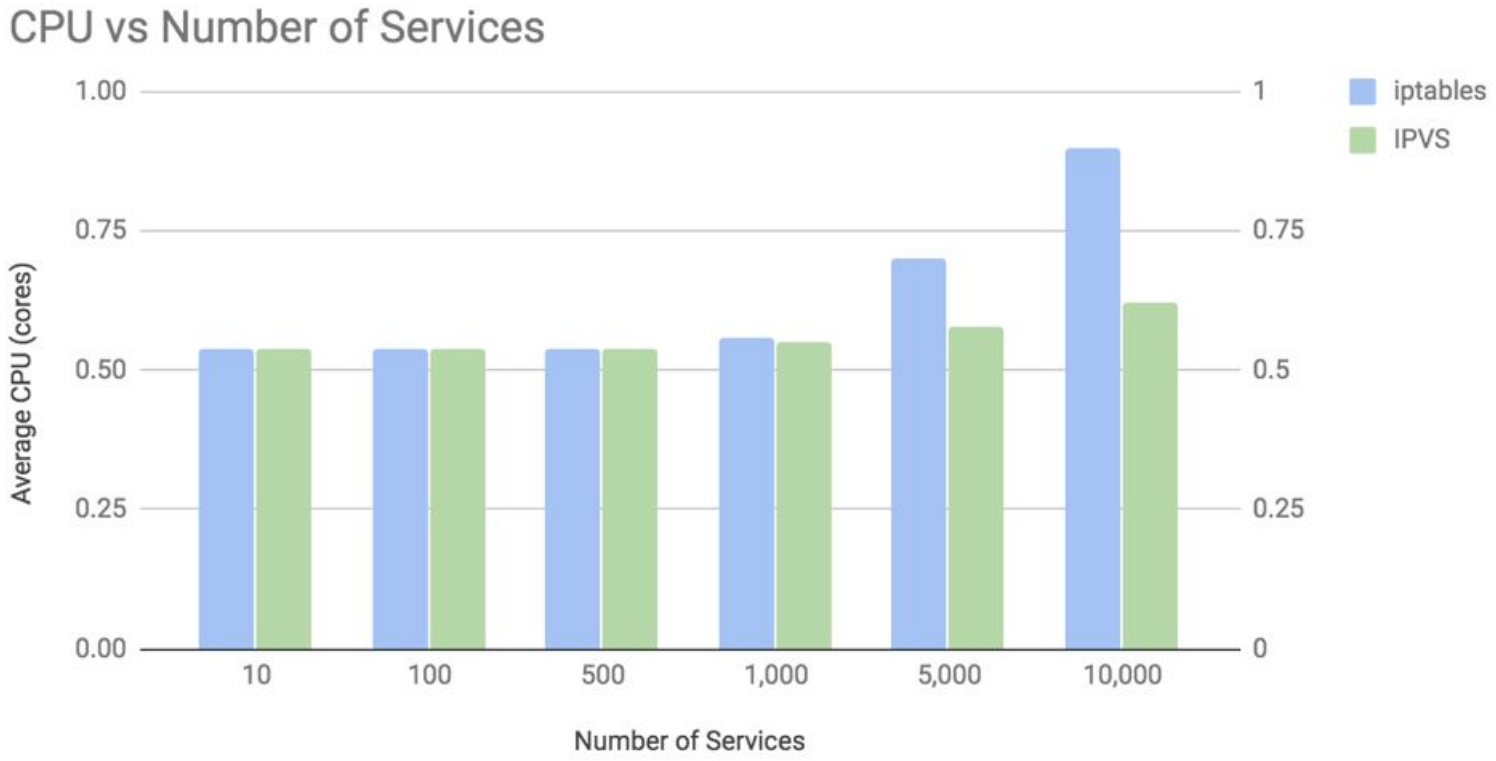
Linux IPVS (IP Virtual Server)

- Favorite of kubernetes for high performance for existing generation
- IPVS is part of LVS (Linux virtual server). That act to L4 network switch and also embed on NetFilter
- Main purpose of IPVS was developed for load balance both tcp/udp in front of computer cluster/farm. It will accept and send traffic to real hosts with single ip address
- When compare with iptables. IPVS give better response time and consume lower cpu resource for operate (Aka support more traffic on same resource)
- IPVS was accept from Kubernetes for choose for support very huge traffic on enterprise kubernetes farm.

Linux IPVS (IP Virtual Server)



Linux IPVS (IP Virtual Server)



Linux IPVS (IP Virtual Server)

- Favorite of kubernetes for high performance for existing generation
- IPVS is part of LVS (Linux virtual server). That act to L4 network switch and also embed on NetFilter
- Main purpose of IPVS was developed for load balance both tcp/udp in front of computer cluster/farm. It will accept and send traffic to real hosts with single ip address
- When compare with iptables. IPVS give better response time and consume lower cpu resource for operate (Aka support more traffic on same resource)
- IPVS was accept from Kubernetes for choose with very huge performance