Lesson 2: Network applications

Van-Linh Nguyen

Fall 2024

國立中正大學
Cybersecurity Lab

Outline

- An example of network application
  1. Encapsulation: Full workflow of a network application
  2. OSI model/TCP/IP Model
  3. Packet header and IP addresses
  4. Develop a networking function in C/C++ and Python
  5. Examine the performance of the network quality
- Popular network applications
  1. HTTP, Mail
  2. Multimedia (SIP, Streaming)
- Popular network models
  1. Peer-to-peer

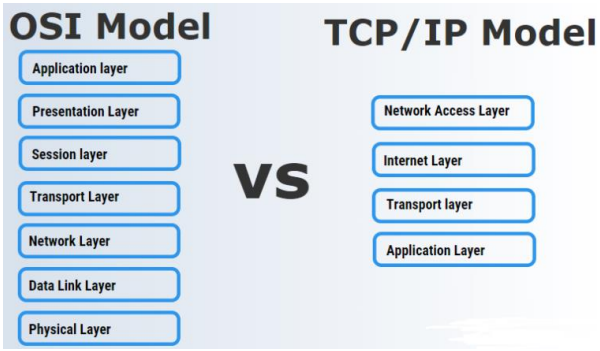
Network application

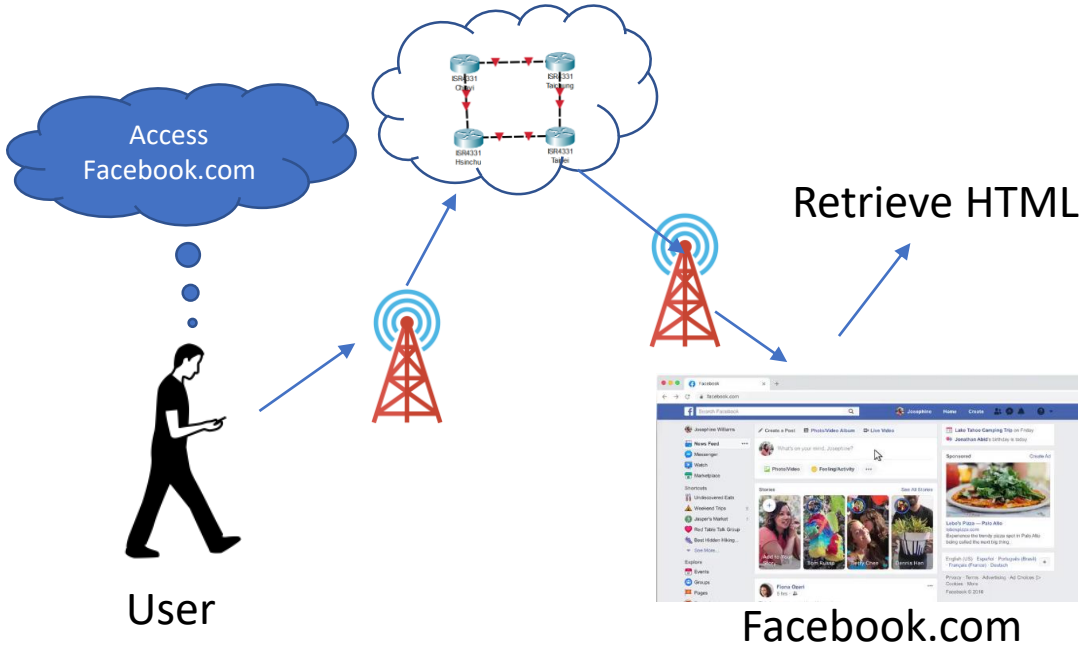
- **Definition**: a software program which operates over a network and allows communication and data sharing between multiple devices and users
  - World Wide Web
  - Email
  - Online Social Network
  - Streaming Audio Video
  - File Sharing
  - Instant Messaging

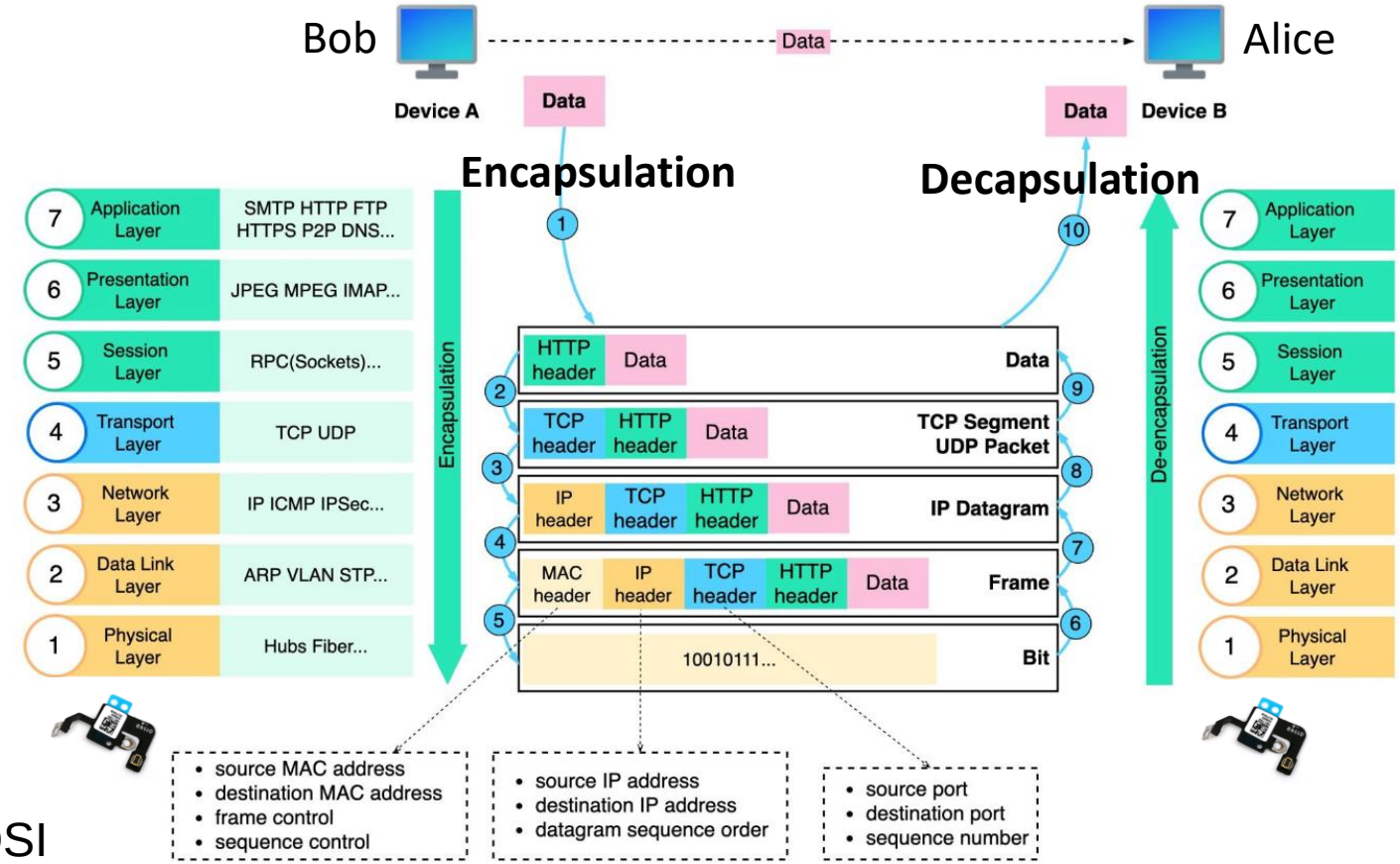
Protocol	Network service	Network application	
		Client	Server
HTTP	WWW	Opera, Firefox, Chrome, Internet Explorer	Apache HTTP Server, NginX HTTP Server, lighttpd, IIS
FTP SMB	File sharing	lftp, atftp, SmartFTP, smbclient	ProFTPD, vsftpd, Pure-FTPD, samba
SQL	Databases	PostgreSQL Client, MySQL Client	PostgreSQL Server, MySQL Server
DHCP SLAAC DNSRA	Network configuration	ISC DHCP Client, The Configurable DHCP Client, Microsoft DHCP Client	ISC DHCP Server, Tiny DHCP Server, Open DHCP Server
RADIUS DIAMETER 802.1X	AAA	FreeRADIUS Client, Radiusclient, Radiusclient-ng, JRadius, Pyrad	FreeRADIUS Server, OpenRadius, Cistron Radius, XTRadius
POP IMAP SMTP	e-mail	Outlook, Thunderbird, Opera Mail	sendmail, postfix, Exchange, courier

Credit to Wojciech Guminski et al.

An example





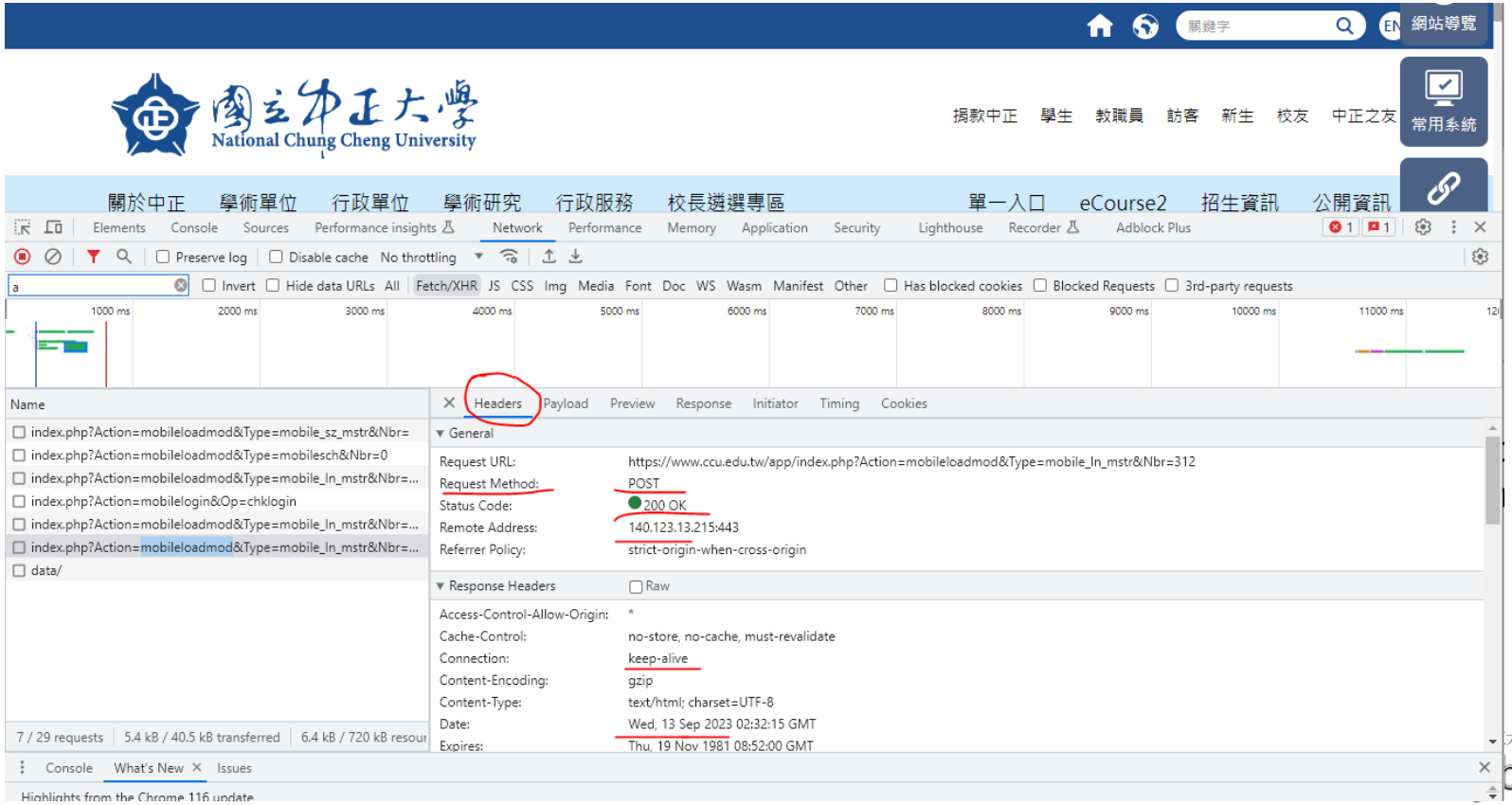


TCP/IP Model is more popular in industry than OSI
OSI is for academic research

https://blog.bytebytego.com/

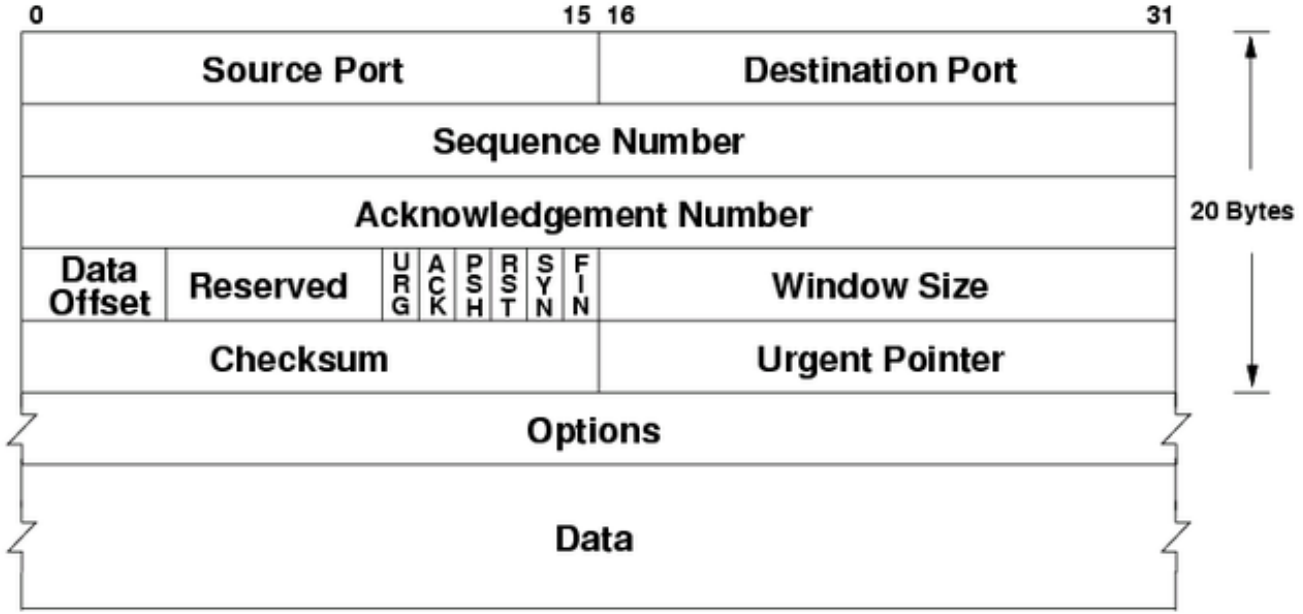
What is HTTP header?

- The HTTP headers are used to pass additional information between the clients and the server through the request and response header



What is TCP header?

- The Transmission Control Protocol (TCP) header is the first 24 bytes of a TCP segment that contains the parameters and state of an end-to-end TCP socket



What is an Internet Protocol (IP)

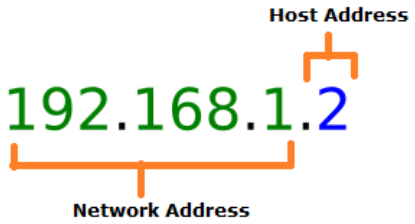
- Definition: the unique identifying number assigned to every device connected to the internet

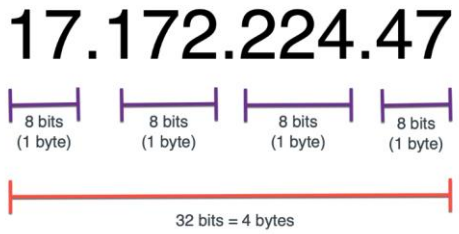
- IPv4 (32 bits): 192.168.1.2

(max 4.3 billion addresses)

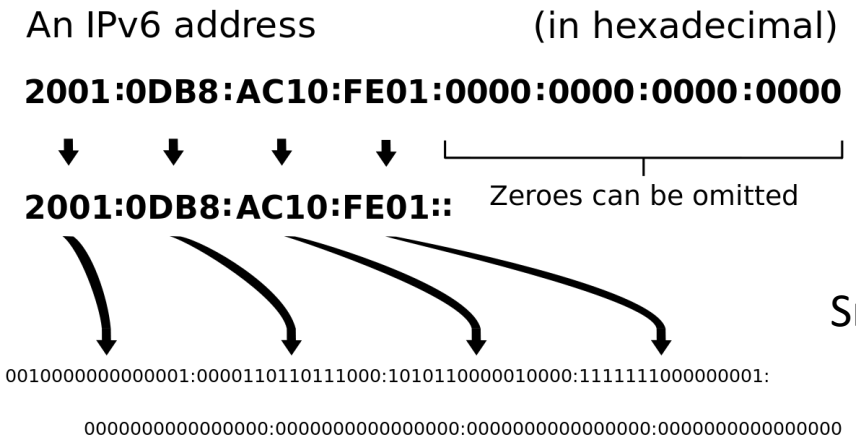
- IPv6 (128 bits)

(max 7.9 x 10^{28} addresses)





Src: Samsung



Src: wikipedia

IP classes

00000000. 00000000 . 00000000. 00000000
11111111. 00000000 . 00000000. 00000000
2^{24} ~ 16million

11111111. 11111111. 00000000. 00000000
2^{16} ~ 64k

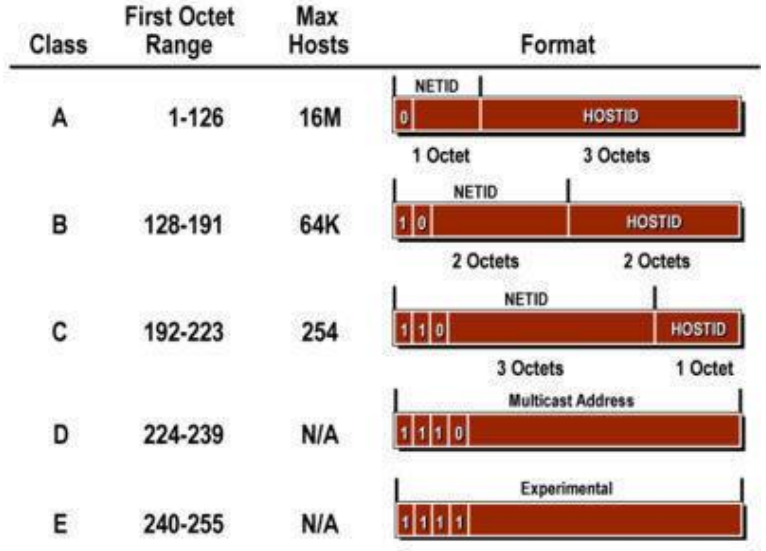
- There are five different IP classes

Since IPv4 is exceeded, we need to transfer to IPv6

- We often use Class A-C

Five Different Classes of IPv4 Addresses

Class	First Octet decimal (range)	First Octet binary (range)	IP range	Subnet Mask	Hosts per Network ID	# of networks
Class A	0 – 127	0XXXXXXX	0.0.0.0-127.255.255.255	255.0.0.0	$2^{24}-2$	2^{7}
Class B	128 – 191	10XXXXXX	128.0.0.0-191.255.255.255	255.255.0.0	$2^{16}-2$	2^{14}
Class C	192 – 223	110XXXXX	192.0.0.0-223.255.255.255	255.255.255.0	$2^{8}-2$	2^{21}
Class D (Multicast)	224 – 239	1110XXXX	224.0.0.0-239.255.255.255			
Class E (Experimental)	240 – 255	1111XXXX	240.0.0.0-255.255.255.255			

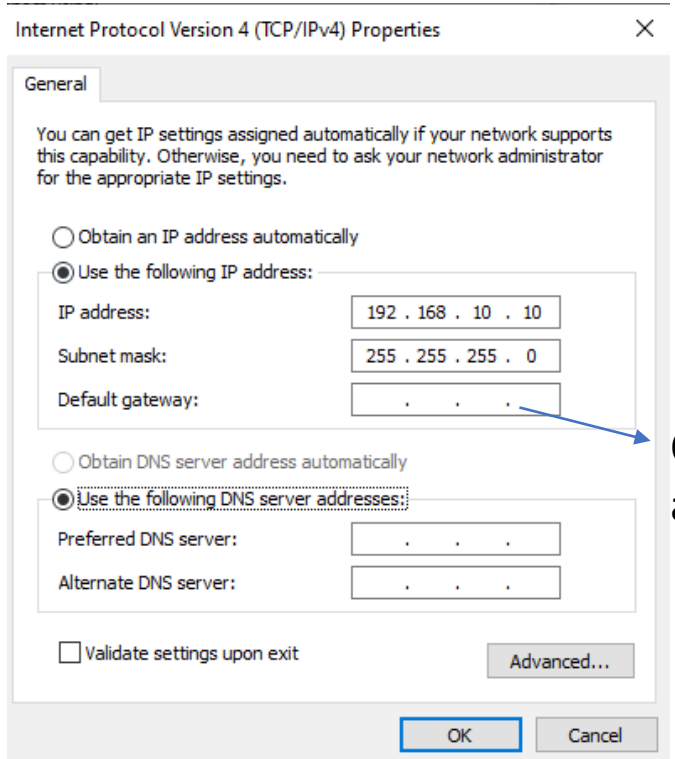


$$111111111 = 2^7 + 2^6 + ... + 2^0 = 255$$

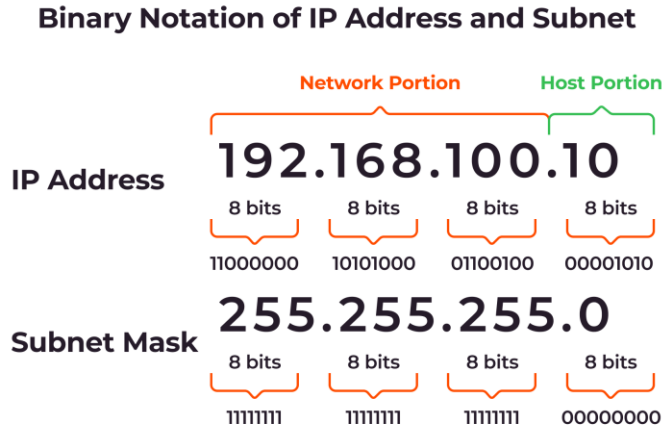
What is a subnet mask?

- The subnet mask splits the IP address into the host and network addresses
- To define which part of the IP address belongs to the device and which part belongs to the network

Network Bits	Subnet Mask	Bits Borrowed	Subnets	Hosts/Subnet
16	255.255.0.0	0	0	65534
17	255.255.128.0	1	2	32766
18	255.255.192.0	2	4	16382
19	255.255.224.0	3	8	8190
20	255.255.240.0	4	16	4094
21	255.255.248.0	5	32	2046
22	255.255.252.0	6	64	1022
23	255.255.254.0	7	128	510
24	255.255.255.0	8	256	254
25	255.255.255.128	9	512	126
26	255.255.255.192	10	1024	62
27	255.255.255.224	11	2048	30
28	255.255.255.240	12	4096	14
29	255.255.255.248	13	8192	6
30	255.255.255.252	14	16384	2



Our router address



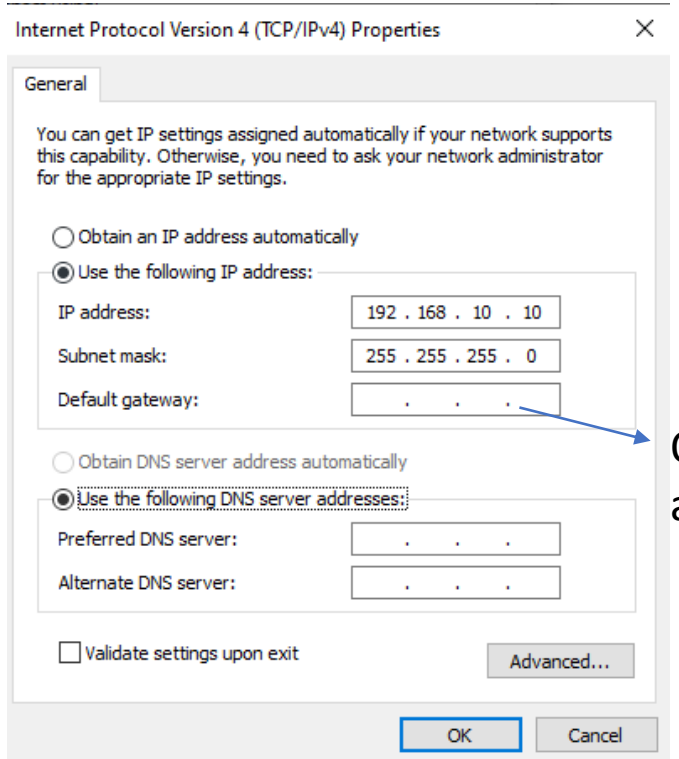
8 bits: 2^8 = 256 hosts

國立中正大學 Cybersecurity Lab

$$111111111 = 2^7 + 2^6 + ... + 2^0 = 255$$

What is a subnet mask?

- The subnet mask splits the IP address into the host and network addresses
- To define which part of the IP address belongs to the device and which part belongs to the network



Our router address

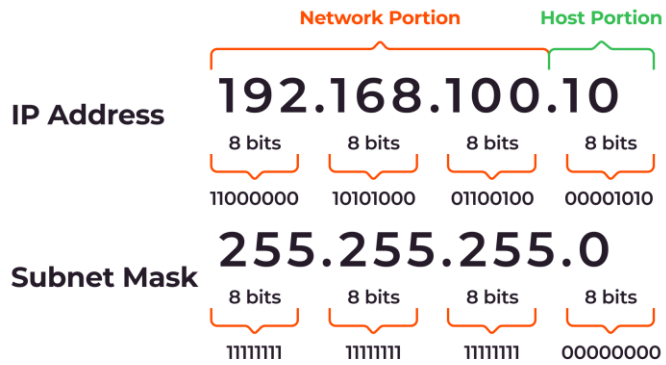
Class B

Network Bits	Subnet Mask	Bits Borrowed	Subnets	Hosts/Subnet
16	255.255.0.0	0	0	65534
17	255.255.128.0	1	2	32766
18	255.255.192.0	2	4	16382
19	255.255.224.0	3	8	8190
20	255.255.240.0	4	16	4094
21	255.255.248.0	5	32	2046
22	255.255.252.0	6	64	1022
23	255.255.254.0	7	128	510
24	255.255.255.0	8	256	254
25	255.255.255.128	9	512	126
26	255.255.255.192	10	1024	62
27	255.255.255.224	11	2048	30
28	255.255.255.240	12	4096	14
29	255.255.255.248	13	8192	6
30	255.255.255.252	14	16384	2

Class C

Network Bits	Subnet Mask	Bits Borrowed	Subnets	Hosts/Subnet
24	255.255.255.0	0	1	254
25	255.255.255.128	1	2	126
26	255.255.255.192	2	4	62
27	255.255.255.224	3	8	30
28	255.255.255.240	4	16	14
29	255.255.255.248	5	32	6
30	255.255.255.252	6	64	2

Binary Notation of IP Address and Subnet



8 bits: 2^8 = 256 hosts

國立中正大學 Cybersecurity Lab

Recall the concept of subnetting

- Subnetting

Number of network bits extracted from the host number (bits for subnet ID)

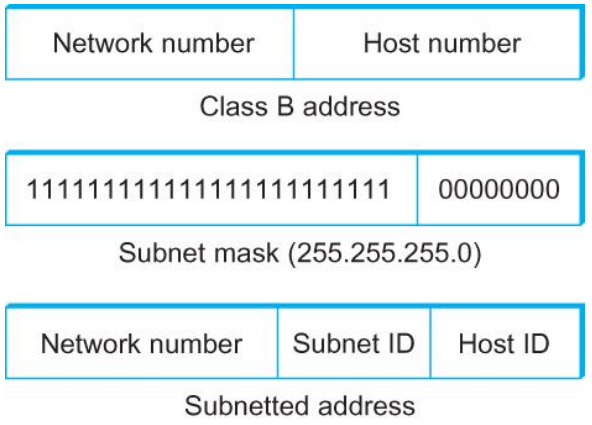
- Number of subnets = 2^N

Number of remaining bits for host number

- Number of hosts per subnet = (2^H) - 2

Class C

Network Bits	Subnet Mask	Bits Borrowed	Subnets	Hosts/Subnet
24	255.255.255.0	0	1	254
25	255.255.255.128	1	2	126
26	255.255.255.192	2	4	62
27	255.255.255.224	3	8	30
28	255.255.255.240	4	16	14
29	255.255.255.248	5	32	6
30	255.255.255.252	6	64	2



A method to find the subnet mask

- Create a Class C network from 192.168.1.0 with 4 subnets and each subnet has max. 62 hosts
- IP : 192.168.1.0
- Class C Subnet: 00000000 → transfer 00 to 11 and add 000000

$2^2 = 4$ $2^6 = 64$

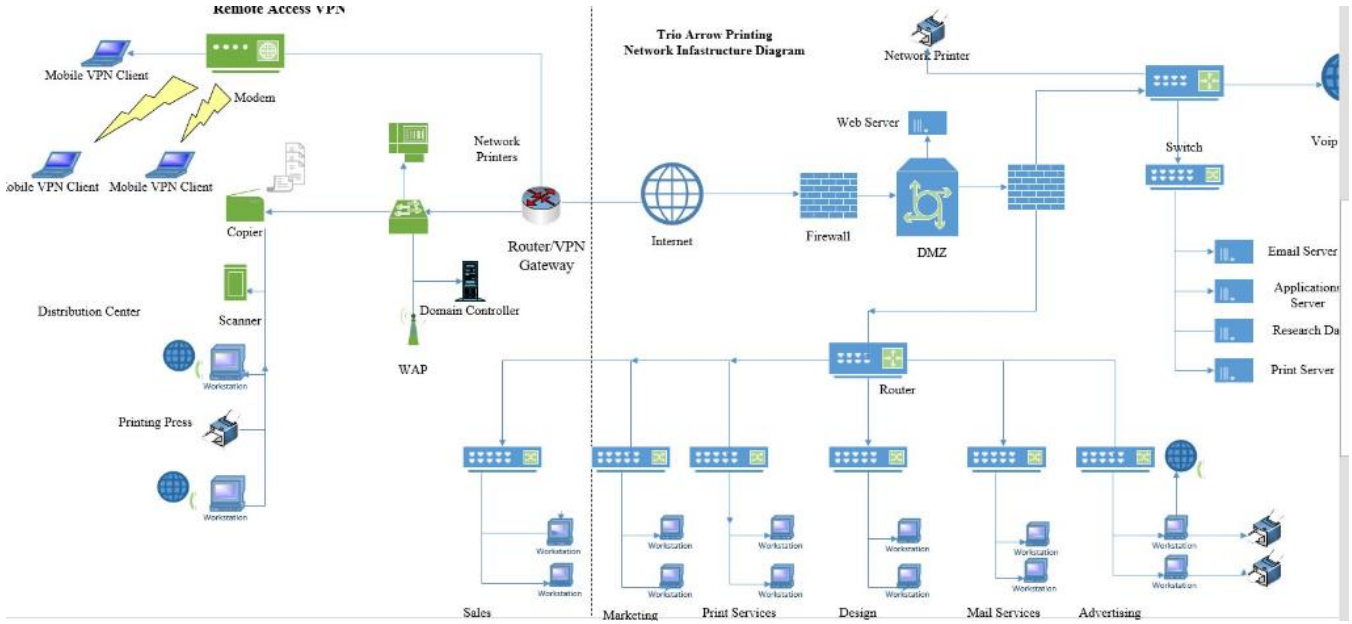
Bit: 11000000 ~ $2^7 + 2^6 = 128+64 = 192$

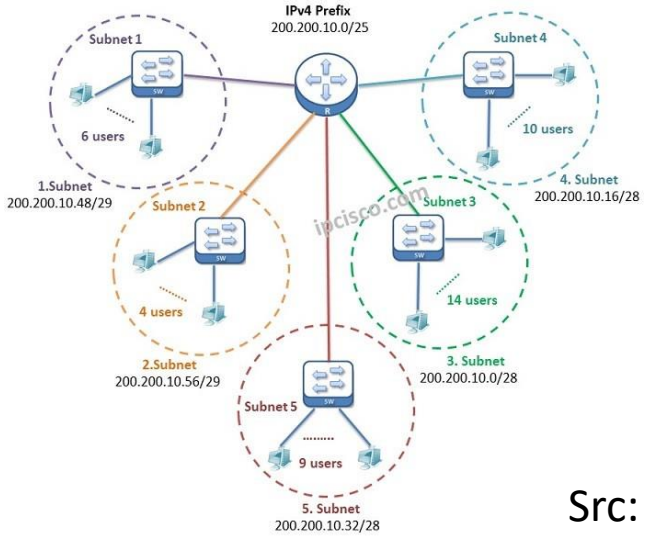
- Subnet: 255.255.255.192

Let us do a simple example

Create a subnet mask for a private network from 192.168.1.0 with max 30 hosts per subnet!

Subnet is important when we want to create different networks within a local area and doesn't allow a network to access each other (the other method is to use VLAN)

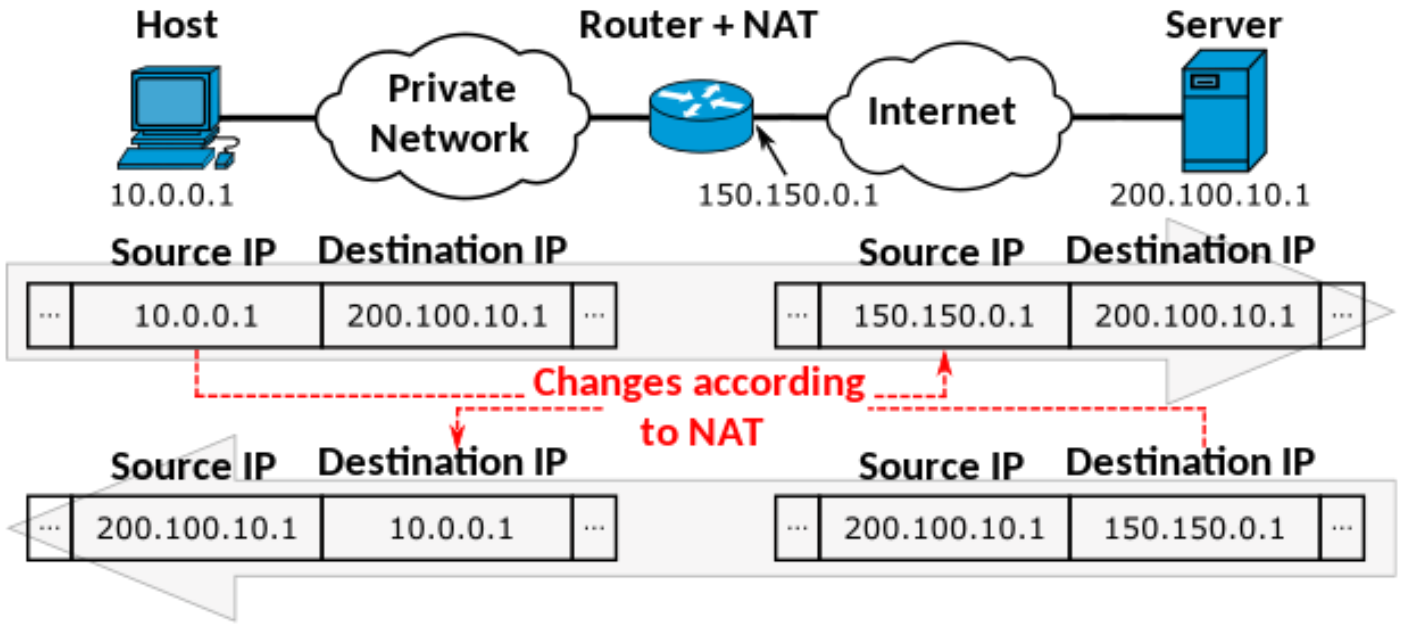




Src: IPcisco

Temporary solution to save IPv4 space

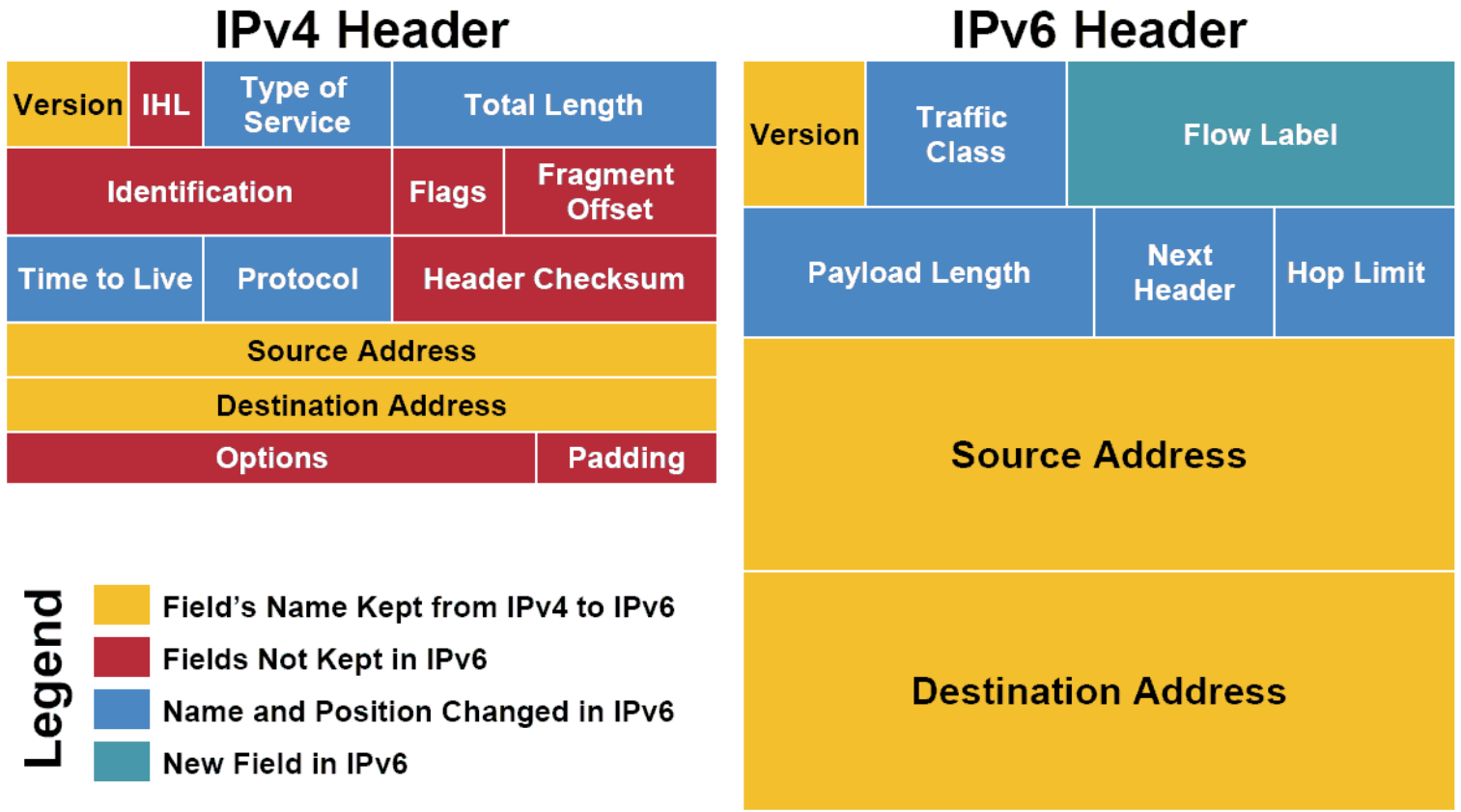
- Use Network Address Translation (NAT): a way to map multiple private addresses inside a local network to a public IP address before transferring the information onto the internet



Src: Wikipedia

IPv4 vs IPv6

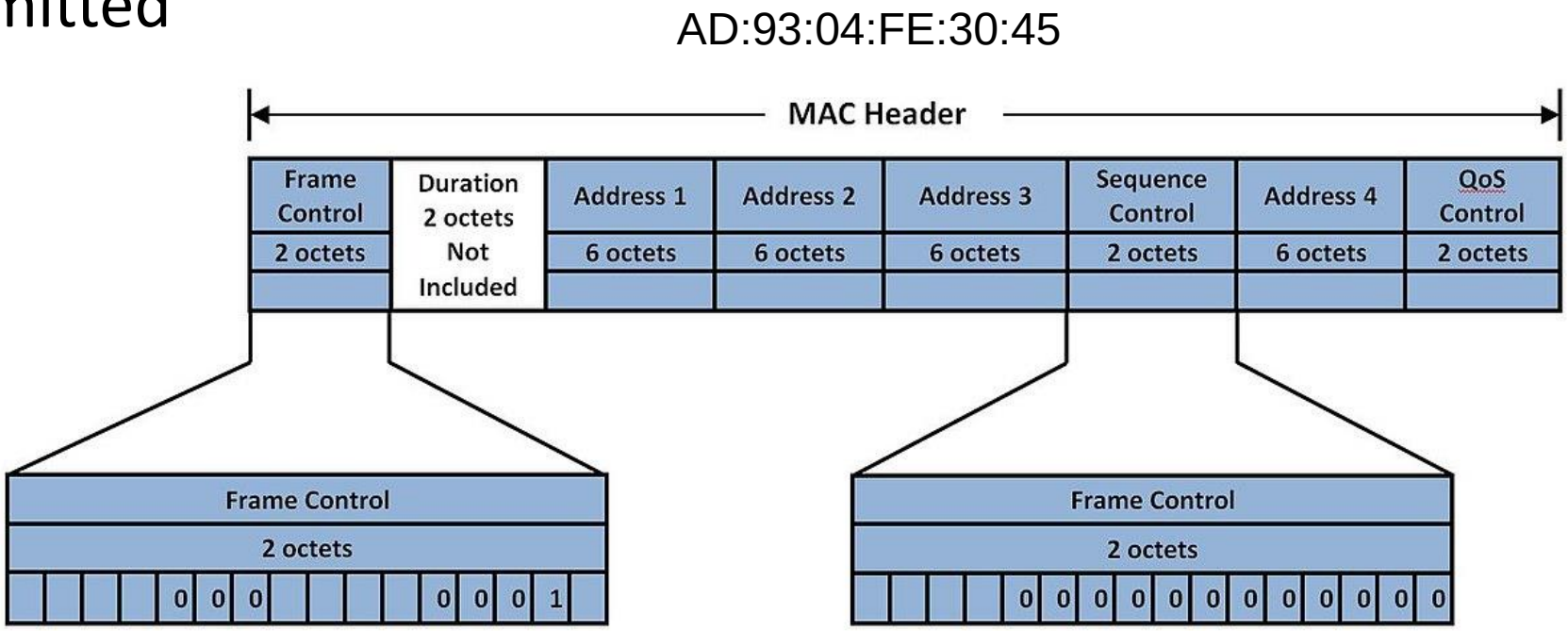
- IPv6 can cover near all devices without worrying out of addresses



https://www.cables-solutions.com/

What is a MAC header? 48 bits

- Media Access Control (MAC) header : The data fields added at the beginning of a network packet in order to turn it into a frame to be transmitted

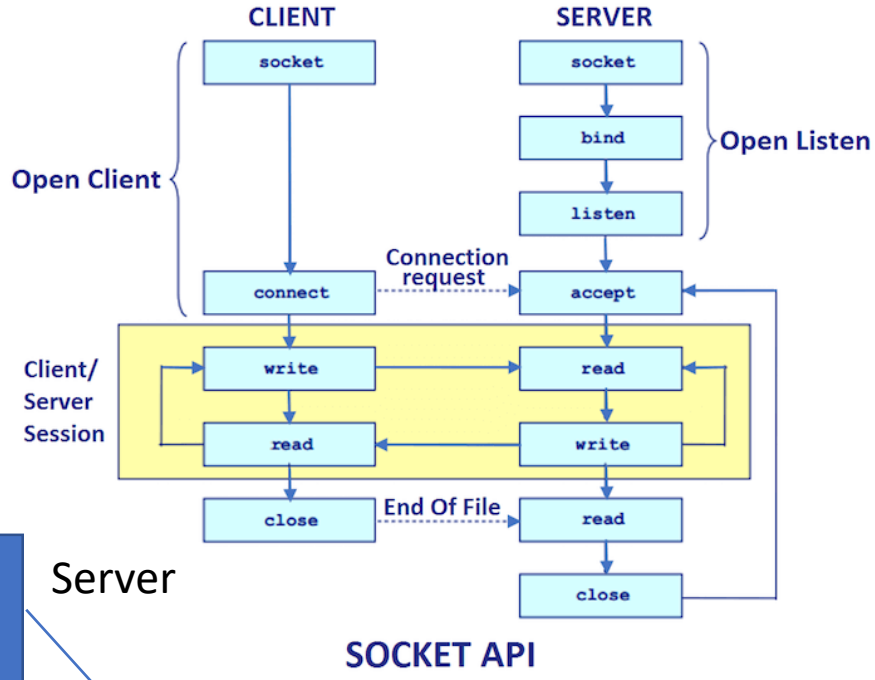


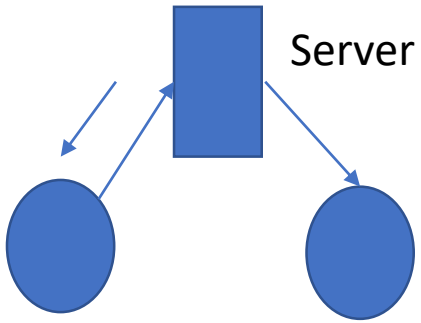
Developing networking functions for the app

- Socket programming: a method to connect two nodes over a network to establish a means of communication between those two nodes

- Basic operations
  1. Creating a socket
  2. Attaching a socket to the network
  3. Sending and receiving messages through the socket
  4. Closing the socket

- Build a chat program (via Internet)





Socket in C/C++

- Socket Family
  - PF_INET denotes the Internet family
  - PF_UNIX denotes the Unix pipe facility
  - PF_PACKET denotes direct access to the network interface (i.e., it bypasses the TCP/IP protocol stack)

- Socket Type
  - SOCK_STREAM is used to denote a byte stream
  - SOCK_DGRAM is an alternative that denotes a message oriented service, such as that provided by UDP

Creating a Socket

```
int sockfd = socket(address_family, type, protocol);
```

- The socket number returned is the socket descriptor for the newly created socket

- `int sockfd = socket (PF_INET, SOCK_STREAM, 0);`
- `int sockfd = socket (PF_INET, SOCK_DGRAM, 0);`

The combination of PF_INET and SOCK_STREAM implies TCP

Client-Server Model with TCP

Server

- Passive open
- Prepares to accept connection, does not actually establish a connection

Server invokes

```
int bind (int socket, struct sockaddr *address,
                                    int addr_len)
int listen (int socket, int backlog)
int accept (int socket, struct sockaddr *address,
                                int *addr_len)
```

Client-Server Model with TCP

Bind

- Binds the newly created socket to the specified address i.e. the network address of the local participant (the server)
- Address is a data structure which combines IP and port

Listen

- Defines how many connections can be pending on the specified socket

Client-Server Model with TCP

Accept

- Carries out the passive open
- Blocking operation
  - Does not return until a remote participant has established a connection
  - When it does, it returns a new socket that corresponds to the new established connection and the address argument contains the remote participant's address

Client-Server Model with TCP

Client

- Application performs active open
- It says who it wants to communicate with

Client invokes

```
int connect (int socket, struct sockaddr *address,
                                    int addr_len)
```

Connect

- Does not return until TCP has successfully established a connection at which application is free to begin sending data
- Address contains remote machine's address

Client-Server Model with TCP

In practice

- The client usually specifies only remote participant's address and let's the system fill in the local information
- Whereas a server usually listens for messages on a well-known port
- A client does not care which port it uses for itself, the OS simply selects an unused one

Client-Server Model with TCP

Once a connection is established, the application process invokes two operation

```
int send (int socket, char *msg, int msg_len,
                                    int flags)

int recv (int socket, char *buff, int buff_len,
                                       int flags)
```

Example Application: Server

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#define SERVER_PORT 5432
#define MAX_PENDING 5
#define MAX_LINE 256
int main()
{
        struct sockaddr_in sin;
        char buf[MAX_LINE];
        int len;
        int s, new_s;
        /* build address data structure */
        bzero((char *)&sin, sizeof(sin));
        sin.sin_family = AF_INET;
        sin.sin_addr.s_addr = INADDR_ANY;
        sin.sin_port = htons(SERVER_PORT);

        /* setup passive open */
        if ((s = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
                        perror("simplex-talk: socket");
                        exit(1);
        }
```

Example Application: Client

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#define SERVER_PORT 5432
#define MAX_LINE 256
int main(int argc, char * argv[])
{
        FILE *fp;
        struct hostent *hp;
        struct sockaddr_in sin;
        char *host;
        char buf[MAX_LINE];
        int s;
        int len;
        if (argc==2) {
                        host = argv[1];
        }
```

Example Application: Client

```
/* translate host name into peer's IP address */
hp = gethostbyname(host);
if (!hp) {
        fprintf(stderr, "simplex-talk: unknown host: %s\n", host);
        exit(1);
}
/* build address data structure */
bzero((char *)&sin, sizeof(sin));
sin.sin_family = AF_INET;
bcopy(hp->h_addr, (char *)&sin.sin_addr, hp->h_length);
sin.sin_port = htons(SERVER_PORT);
/* active open */
if ((s = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
        perror("simplex-talk: socket");
        exit(1);
}
if (connect(s, (struct sockaddr *)&sin, sizeof(sin)) < 0) {
        perror("simplex-talk: connect");
        close(s);
        exit(1);
}
/* main loop: get and send lines of text */
while (fgets(buf, sizeof(buf), stdin)) {
        buf[MAX_LINE-1] = '\0';
        len = strlen(buf) + 1;
        send(s, buf, len, 0);
}
}
```

Example Application: Server

```
        if ((bind(s, (struct sockaddr *)&sin, sizeof(sin))) < 0) {
                perror("simplex-talk: bind");
                exit(1);
        }
        listen(s, MAX_PENDING);
        /* wait for connection, then receive and print text */
        while(1) {
                if ((new_s = accept(s, (struct sockaddr *)&sin, &len)) < 0) {
                perror("simplex-talk: accept");
                exit(1);
        }
        while (len = recv(new_s, buf, sizeof(buf), 0))
                fputs(buf, stdout);
                close(new_s);
        }
}
```

Example Application 2: Socket programing in Python

 # echo-server.py

```
import socket

HOST = "127.0.0.1"  # Standard loopback interface address (localhost)
PORT = 65432  # Port to listen on (non-privileged ports are > 1023)

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    conn, addr = s.accept()
    with conn:
        print(f"Connected by {addr}")
        while True:
            data = conn.recv(1024)
            if not data:
                break
            conn.sendall(data)
```

 # echo-client.py

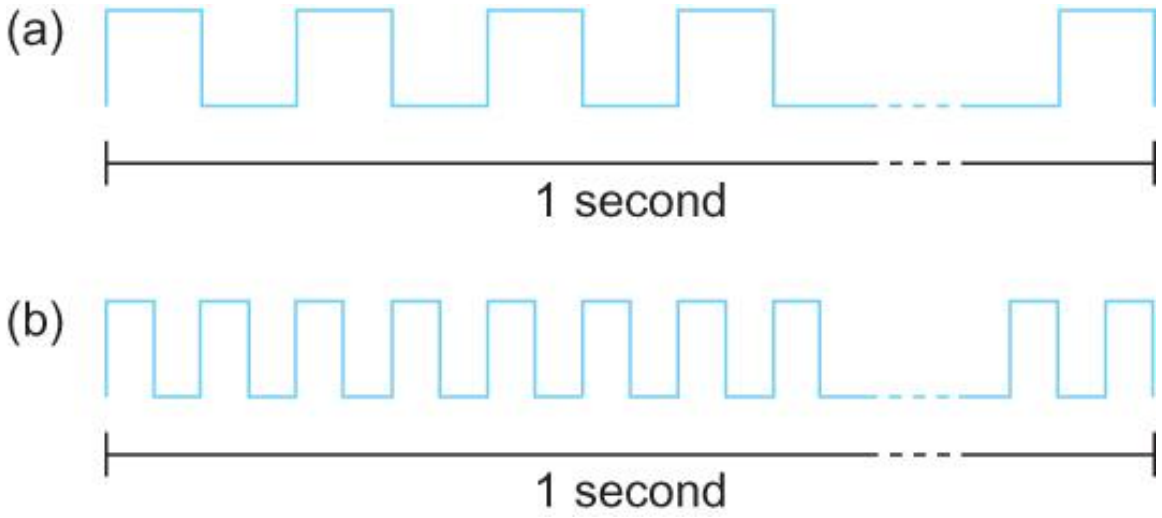
```
import socket

HOST = "127.0.0.1"  # The server's hostname or IP address
PORT = 65432  # The port used by the server

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    s.sendall(b"Hello, world")
    data = s.recv(1024)

print(f"Received {data!r}")
```

Printer Drivers

Examine the performance of the network quality

- Bandwidth
  - Width of the frequency band
  - Number of bits per second that can be transmitted over a communication link
- 1 Mbps: 1×10^6 bits/second = 1×2^{20} bits/sec
- 1×10^{-6} seconds to transmit each bit or imagine that a timeline, now each bit occupies 1 micro second space.
- On a 2 Mbps link the width is 0.5 micro second.
- Smaller the width more will be transmission per unit time.

Bandwidth

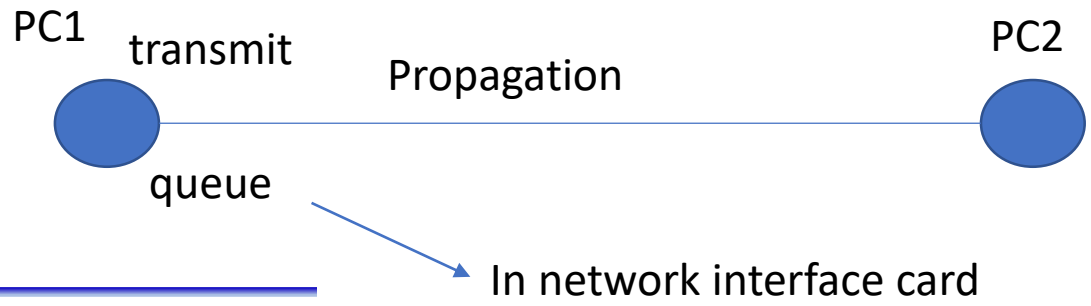


Bits transmitted at a particular bandwidth can be regarded as having some width:

(a) bits transmitted at 1Mbps (each bit 1 μs wide);

(b) bits transmitted at 2Mbps (each bit 0.5 μs wide).

Performance



- Latency = Propagation + transmit + queue
- Propagation = distance/speed of light
- Transmit = size/bandwidth

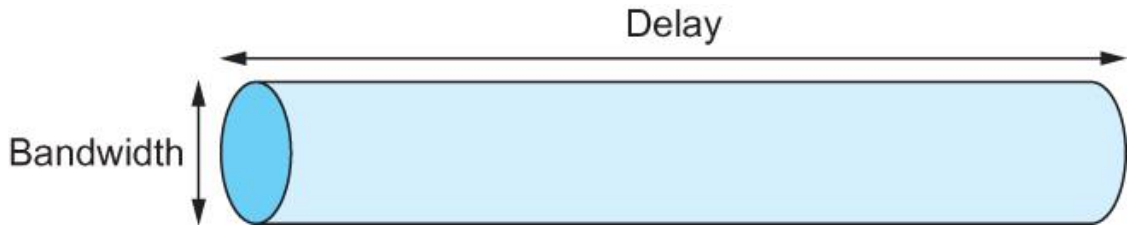
- One bit transmission => propagation is important
- Large bytes transmission => bandwidth is important

Delay vs Bandwidth

- We think the channel between a pair of processes as a hollow pipe
- Latency (delay) length of the pipe and bandwidth the width of the pipe
- Delay of 50 ms and bandwidth of 45 Mbps

$\Rightarrow 50 \times 10^{-3}$ seconds x 45 x 10^{6} bits/second

$\Rightarrow 2.25 \times 10^{6}$ bits = 280 KB data.

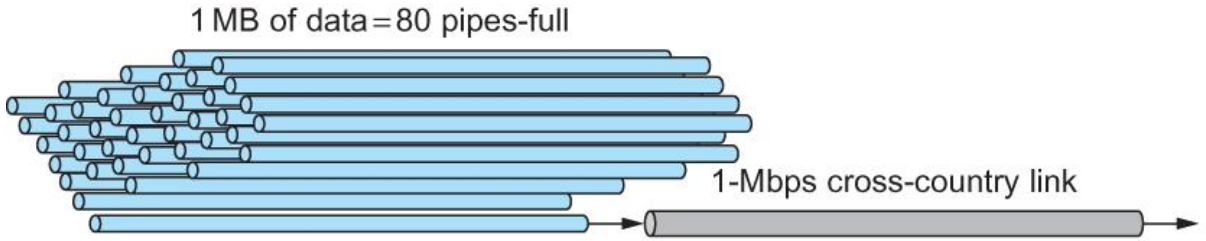


Network as a pipe

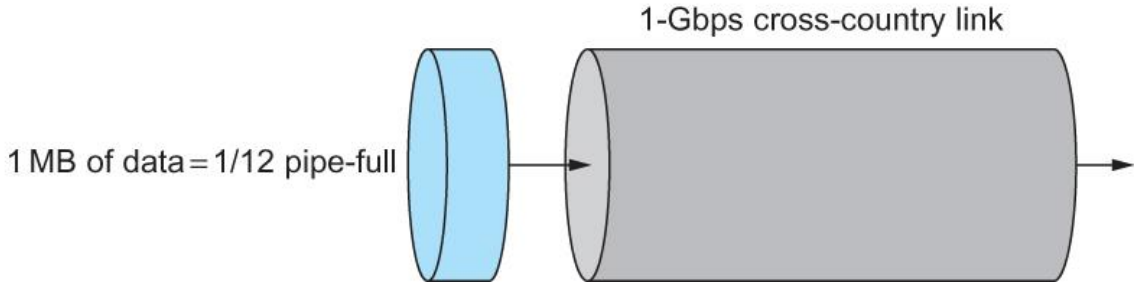
Delay vs Bandwidth

- Relative importance of bandwidth and latency depends on application
  - For large file transfer, bandwidth is critical
  - For small messages (HTTP, NFS, etc.), latency is critical
  - Variance in latency (jitter) can also affect some applications (*e.g.*, audio/video conferencing)

Relationship between bandwidth and latency



Which transmission is faster?



A 1-MB file would fill the 1-Mbps link 80 times,

but only fill the 1-Gbps link 1/12 of one time

Some reasons impact network reliability

- Bits are lost
  - Bit errors (1 to a 0, and vice versa)
  - Burst errors – several consecutive errors
- Packets are lost (Congestion)
- Links and Node failures
- Messages are delayed
- Messages are delivered out-of-order
- Third parties eavesdrop

Popular network applications

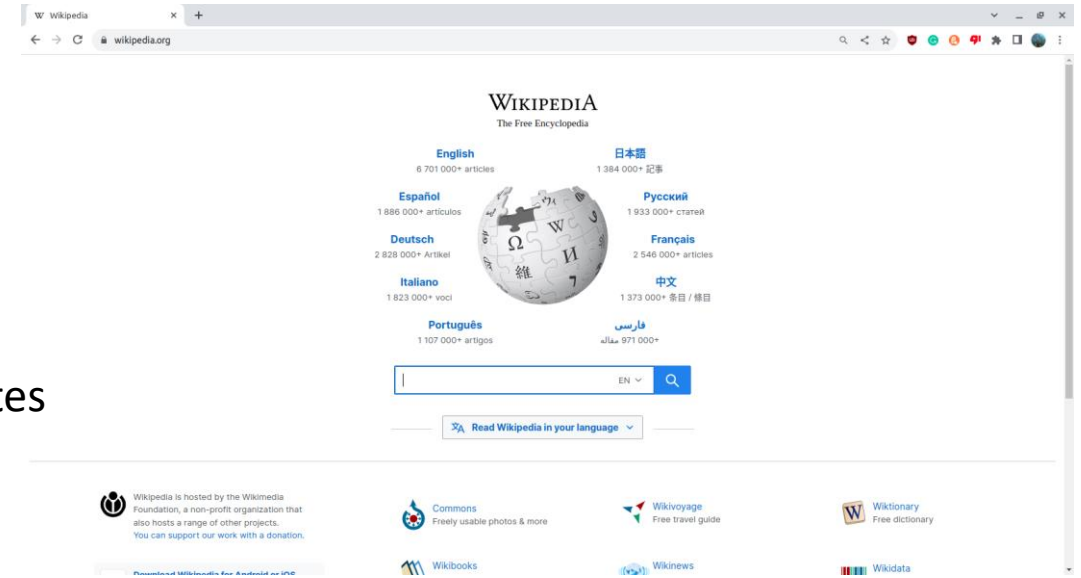
World Wide Web (Web)

- **Definition**: is an information system that enables information sharing over the Internet through user-friendly ways

- The Web is as a set of cooperating clients and servers, all of whom speak the same language (protocol): HTTP or HTTPS

Wikipedia.com
Facebook.com
Twitter.com
Ccu.edu.tw

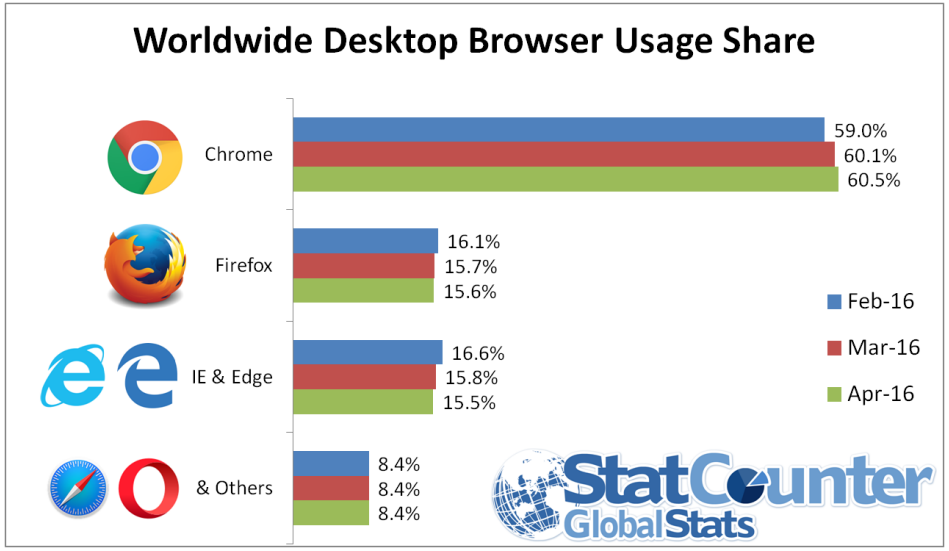
are domain names of famous websites

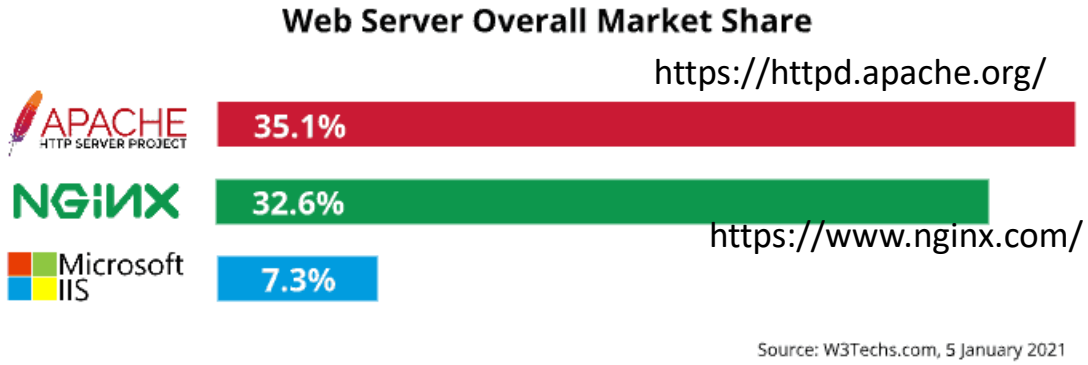


An example of Website

Web browser vs Web Server

- Most people are exposed to the Web through a graphical client program, called Web browser, e.g., **Chrome, Firefox, Safari, Edge**
- The servers providing Web services often run Web server program, e.g., **IIS, Apache, NGINX**





We often install a bundle platform like XAMPP or WAMP

Web URL

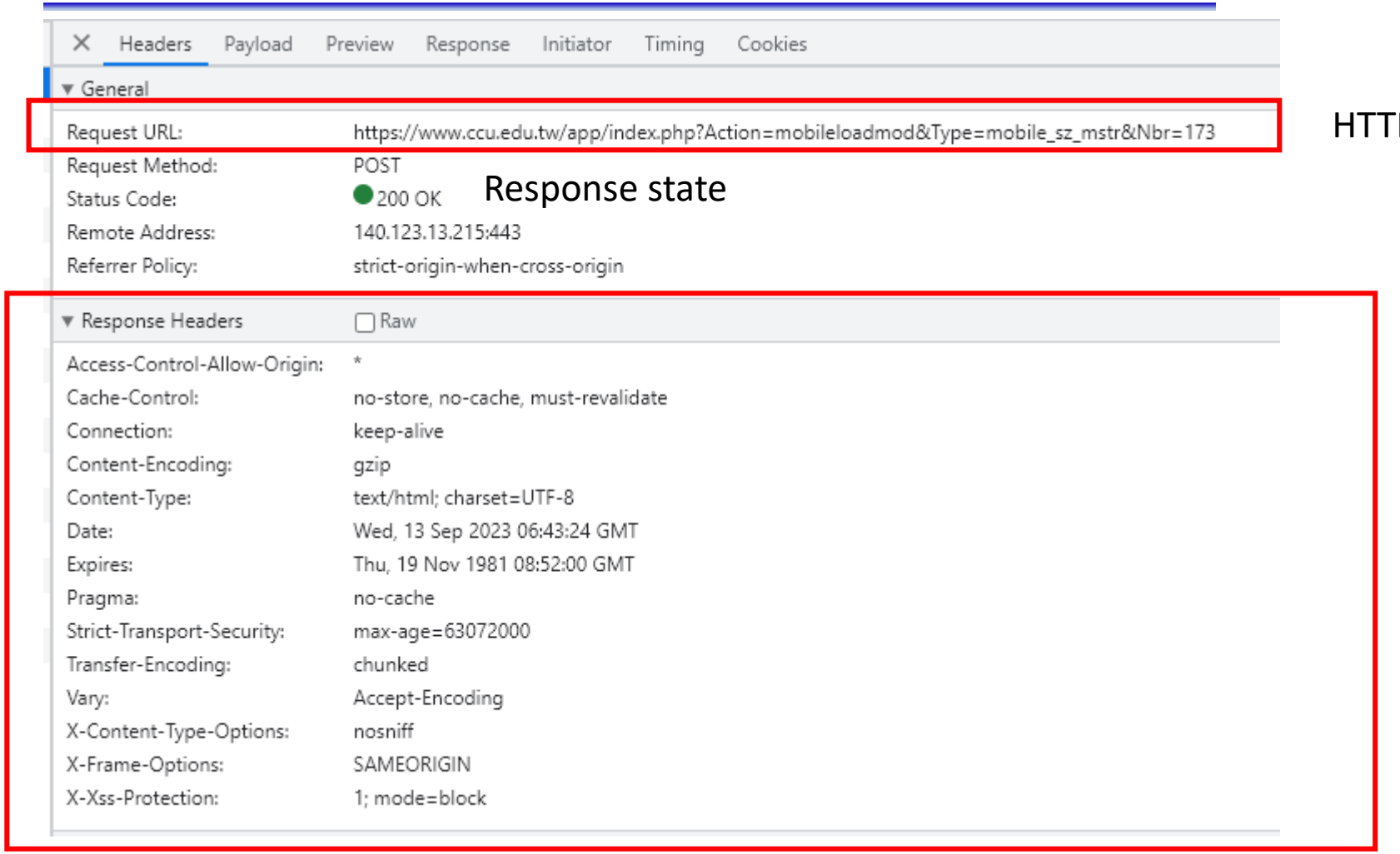
- We often access a resource on the web through putting a special Uniform Resource Locator (URL) on Web browser

Resource on Web

- Most files on the Web contain images and text and many have other objects such as audio and video clips, pieces of code, etc.
- When you ask your browser to view a page, your browser (the client) fetches the page from the server using HTTP running over TCP
- The server will retrieve resources in HyperText Markup Language (HTML) format

You can click View Source on Web browser to see The website's HTML code

HTTP Request



Response state

HTTP Request URL

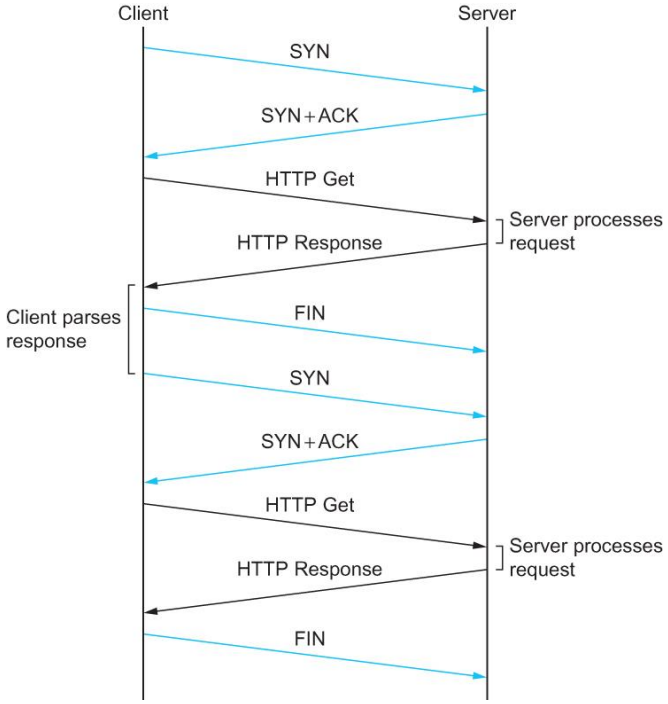
HTTP header

HTTP Response (HTML)

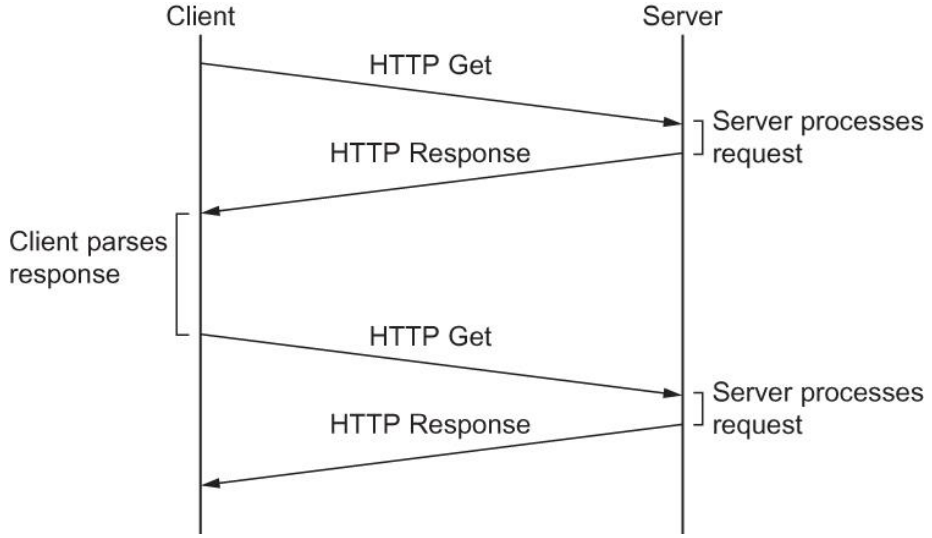
Headers Payload Preview Response Initiator Timing Cookies

```
<div class="module module-adv md_style99">
    <div class="mouter">
    <header class="mt mthide ">


</header>

    <section class="mb">
        <div class="minner">
        <div class="banner mads-li mcarousel owl-carousel __944___listAds">



            <figure class="figBS">

                    <a href="/app/index.php?Plugin=mobile&Action=mobileads&ad=2481" target="_blank" title="駐校藝術家皖嘉博書

                        <img class="img-responsive madv-img madv-img-large-nosmall" src="/var/file/0/1000/randimg/mobileadv



                    </a>
                    <div class="bn-txt bn-bottom"><div class="bn-txt-bg" style="background-color:"></div><div class="bn-mde


            </figure>
```

TCP request in HTTP



HTTP 1.0 behavior

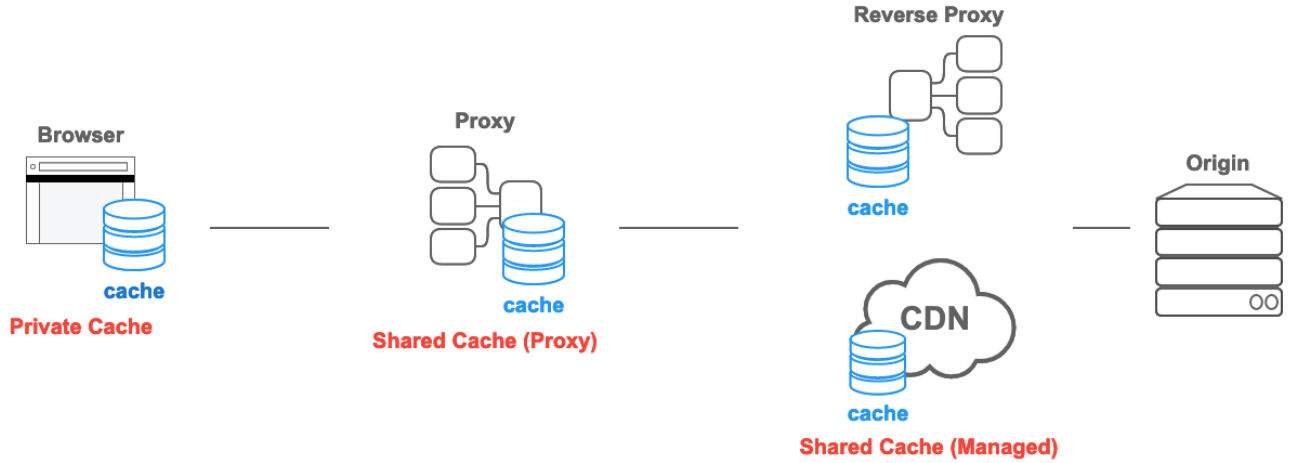


HTTP 1.1 behavior with persistent connections (simpler)

Web Caching and CDN

- **Caching**: retaining HTTP responses and web resources in the cache for the purpose of fulfilling future requests from cache rather than from the origin servers
- Caching has many benefits: From the client's perspective, a page that can be retrieved from a nearby cache can be displayed much more quickly than if it has to be fetched from across the world
- From the server's perspective, having a cache intercept and satisfy a request reduces the load on the server

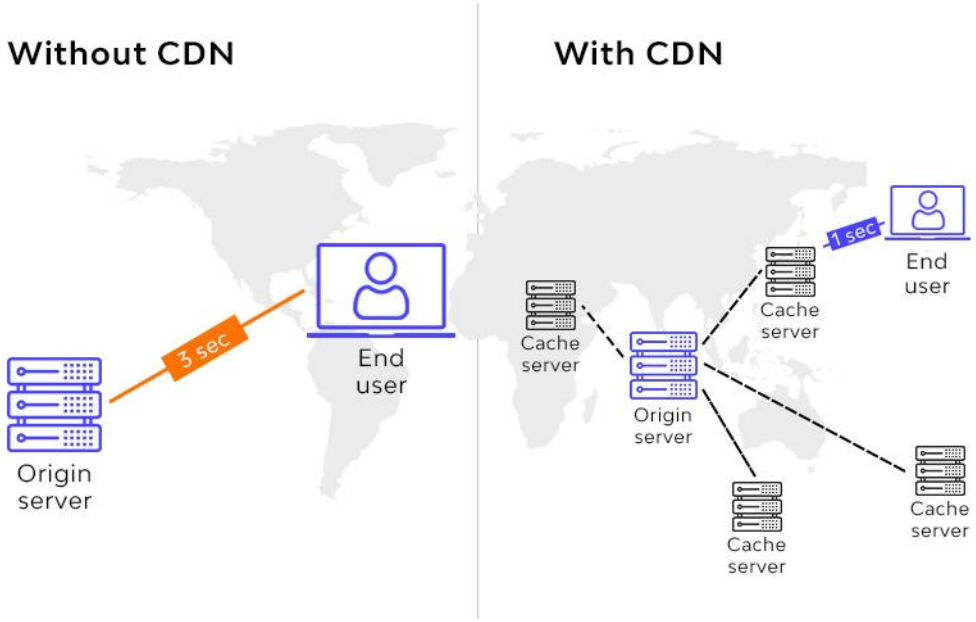
Caching can be implemented in many different places (ISP, web server, web browser)

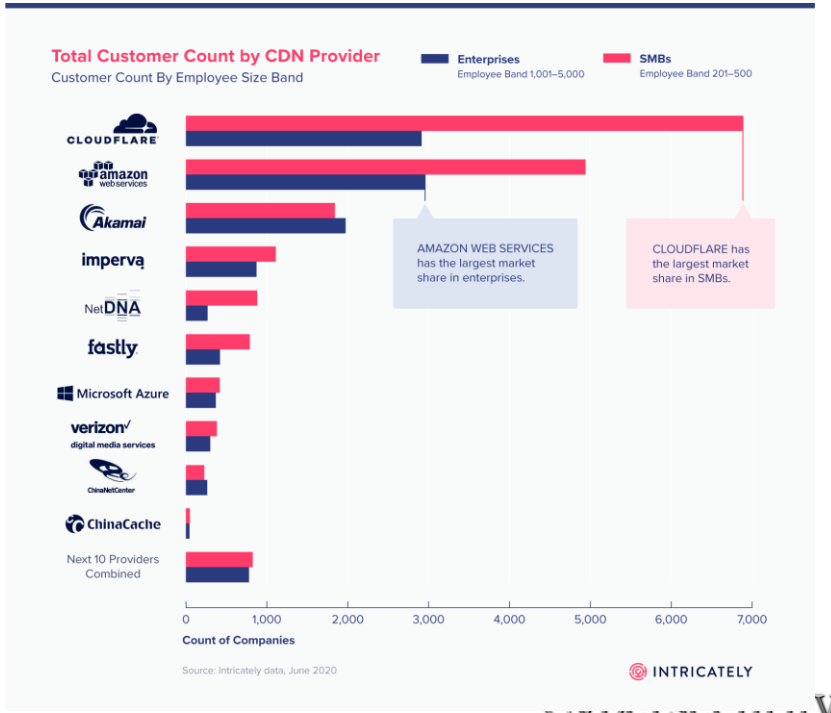


Src: Mozilla

CDN = Caching++

- A content delivery network (CDN) is a distributed network of servers located in different geographic locations around the world. The purpose of a CDN is to improve the performance, reliability, and scalability of delivering web content to users. In addition to caching content, a CDN can also provide other services such as load balancing, security, and analytics



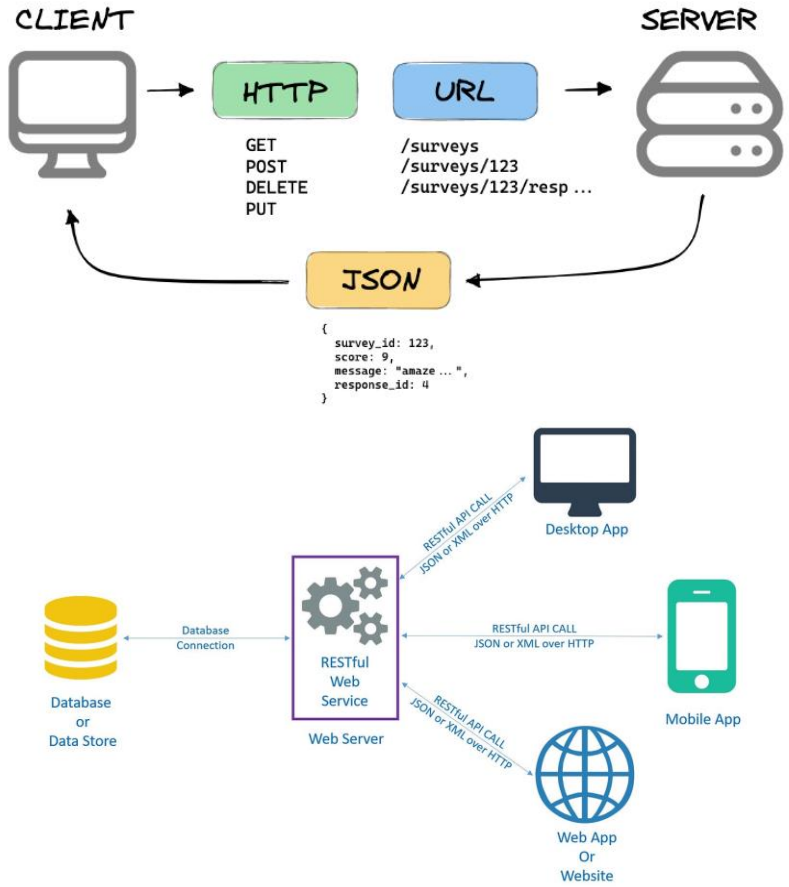


Commercial CDN

- A CDN allows for the quick transfer of assets needed for loading Internet content, including HTML pages, JavaScript files, stylesheets, images, and videos.
- Today the majority of web traffic is served through CDNs, including traffic from major sites like **Facebook, Netflix, and Amazon**

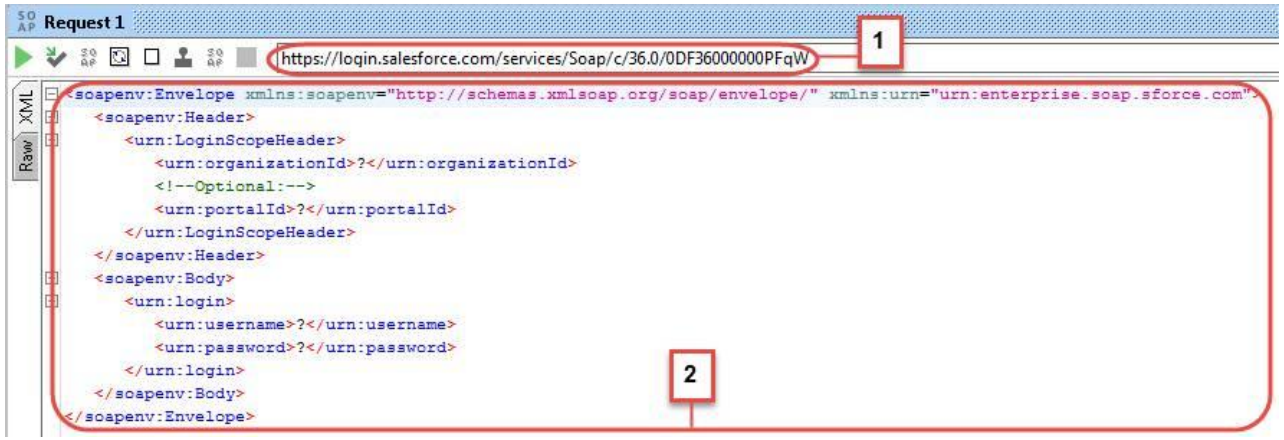
Web API vs JSON

- A Web Application Programming Interface (API) is an application programming interface for the Web
- Web API is designed to help other applications (mobile apps, other websites) to interact with Webserver
- The JavaScript Object Notation (JSON) is widely used for interacting with WebAPI

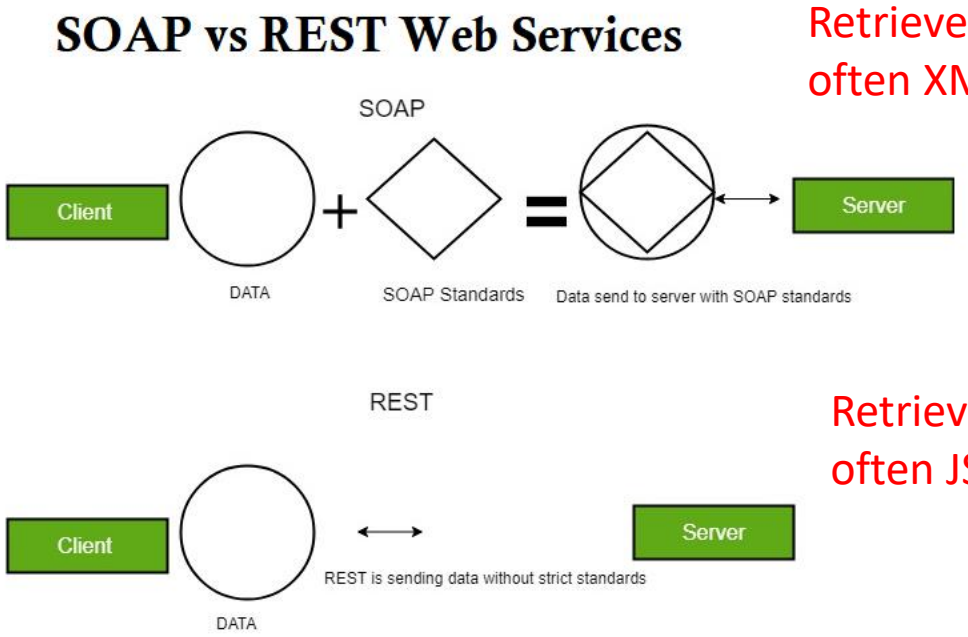


SOAP vs. REST

- Simple Object Access Protocol (SOAP) is a message specification for exchanging information between systems and applications



- REST API is an interface that two computer systems use to exchange information securely over the internet

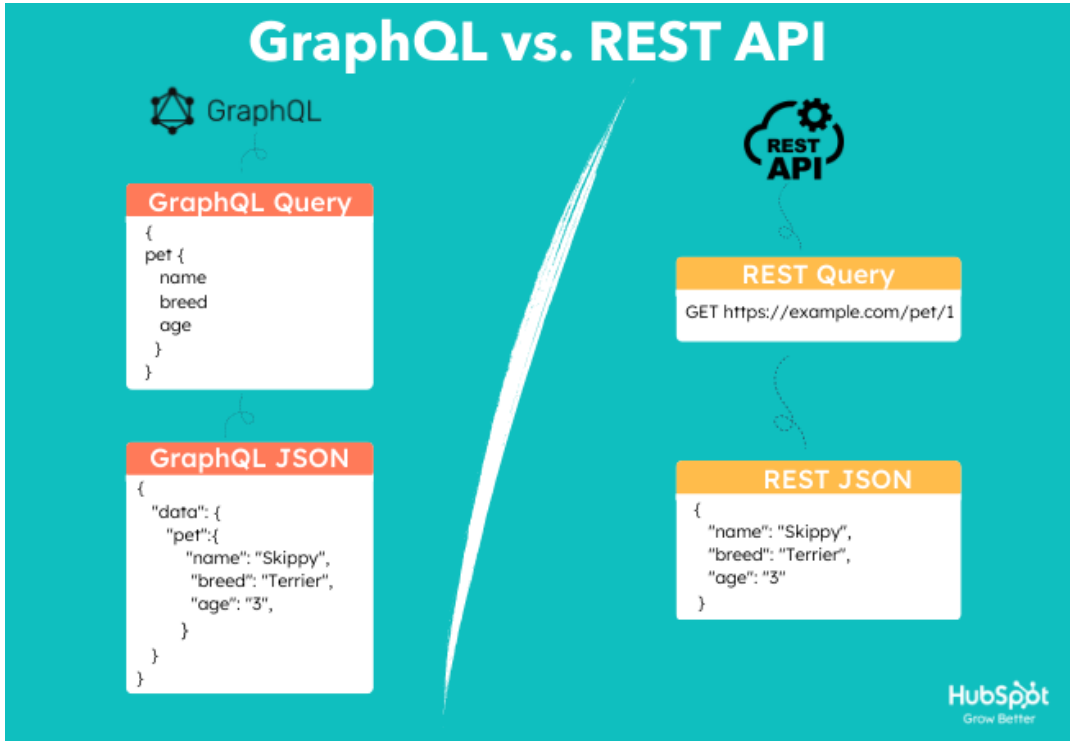


Retrieve data is often XML

Retrieve data is often JSON

REST vs GraphSL

- New technology: GraphSL is an alternative to REST APIs



	GraphQL	REST
Architecture	client-driven	server-driven
Organized in terms of	schema & type system	endpoints
Operations	Query Mutation Subscription	Create, Read, Update, Delete
Data fetching	specific data with a single API call	fixed data with multiple API calls
Community	growing	large
Performance	fast	multiple network calls take up more time
Development speed	rapid	slower
Learning curve	difficult	moderate
Self-documenting	✓	—
File uploading	—	✓
Web caching	(via libraries built on top)	✓
Stability	less error prone, automatic validation and type checking	better choice for complex queries
Use cases	multiple microservices, mobile apps	simple apps, resource-driven apps

Some common HTTP calls & response

- HTTP Request Messages

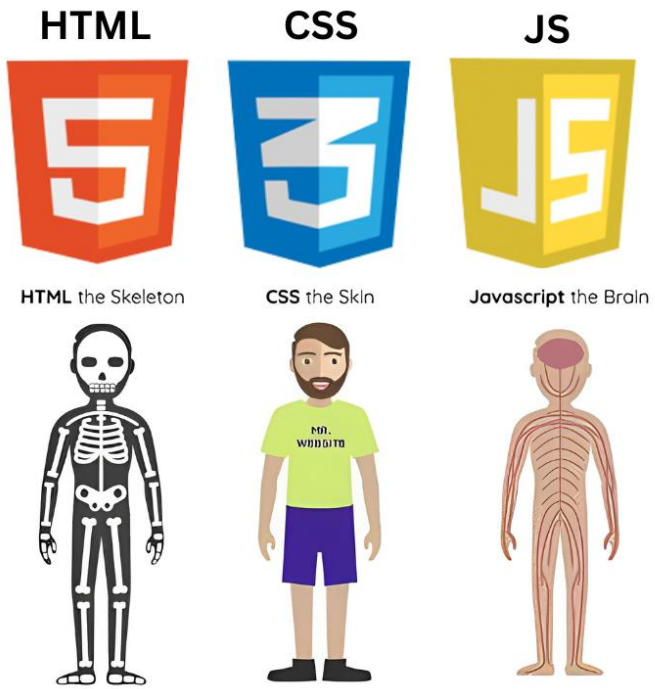
Operation	Description
OPTIONS	Request information about available options
GET	Retrieve document identified in URL
HEAD	Retrieve metainformation about document identified in URL
POST	Give information (e.g., annotation) to server
PUT	Store document under specified URL
DELETE	Delete specified URL
TRACE	Loopback request message
CONNECT	For use by proxies

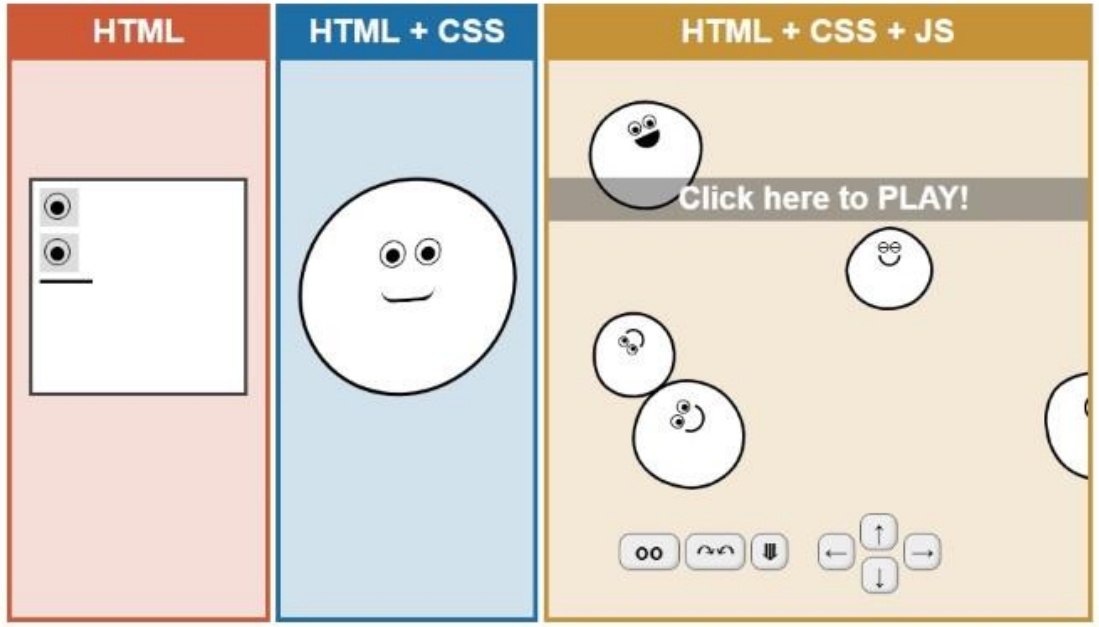
HTTP Response Code

Code	Type	Example Reasons
1xx	Informational	request received, continuing process
2xx	Success	action successfully received, understood, and accepted
3xx	Redirection	further action must be taken to complete the request
4xx	Client Error	request contains bad syntax or cannot be fulfilled
5xx	Server Error	server failed to fulfill an apparently valid request

Programming for client side

- Three popular languages for client side programming: **HTML, Javascript, CSS**





Src: Coco

HTML code

```html
<!DOCTYPE HTML>
<html>

<head>
  <meta http-equiv="Content-Type"
  content="text/html; charset=UTF-8" />
  <title>Hello World</title>
</head>

<body>
  Hello World!
</body>

</html>
```

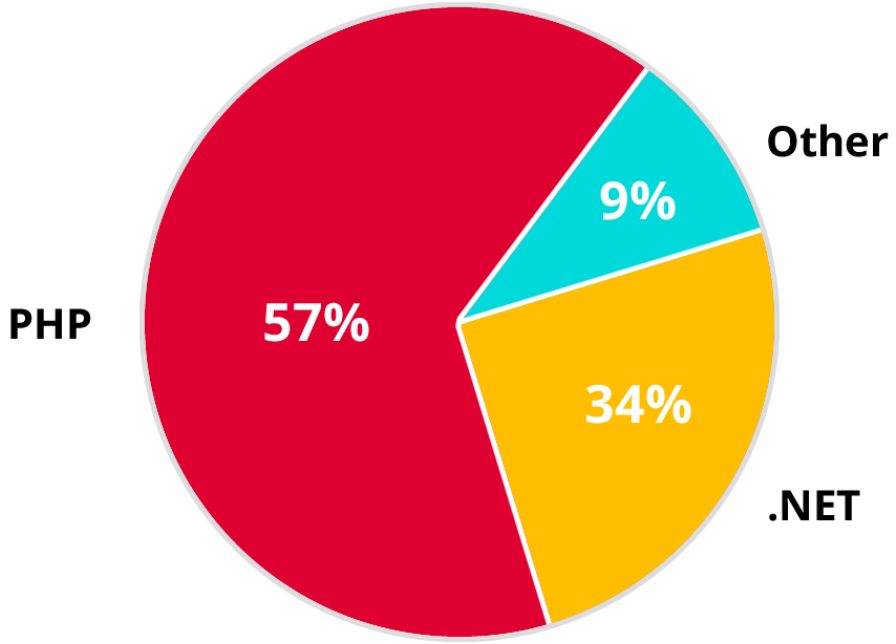
```html
<html>
 <title>Hello World in JavaScript </title>
<body>
  <script>
    document.write("<h2>Hello World in JavaScript </h2>
        ");
    document.write("Prog")
  </script>
 </html>

```

```
/* CSS */
<style>
  .green {
      color: green;
  }
  .blue {
      color: blue;
  }
</style>

<!--html-->
<h1 class="blue green">Hello World!</h1>
```

Server side

- The server program will run special languages:

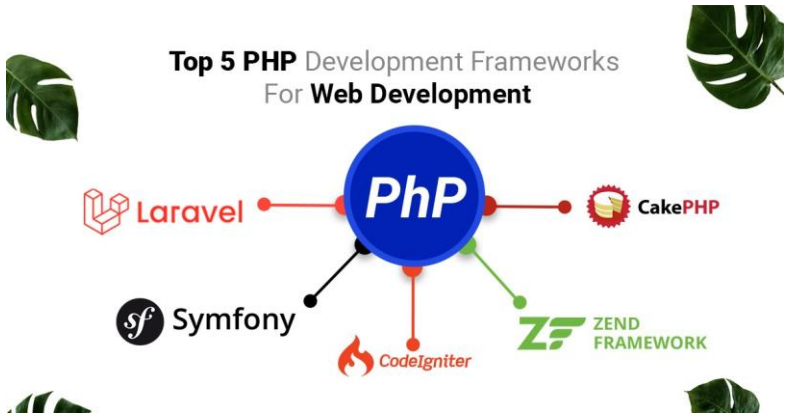
  Apache, NGINX (PHP), IIS (.NET), Oracle (java), Perl, ….



Src: Statista

PHP programming

- Famous platforms



Source: LinkedIn

```php
<?php

namespace App\Command\Web;

use Librarian\Provider\TwigServiceProvider;
use Librarian\WebController;
use Librarian\Response;
use Librarian\Provider\ContentServiceProvider;

class IndexController extends WebController
{
    public function handle()
    {
        /** @var TwigServiceProvider $twig */
        $twig = $this->getApp()->twig;

        $page = 1;
        $limit = $this->getApp()->config->posts_per_page ?: 1
        $params = $this->getRequest()->getParams();

        if (key_exists('page', $params)) {
            $page = $params['page'];
        }

        $start = ($page * $limit) - $limit;

        /** @var ContentServiceProvider $content_provider */
        $content_provider = $this->getApp()->content;
```

.NET programming

- Famous platforms


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.EntityFrameworkCore;
using CipherServices.Services;
using CipherServices.Data;
using CipherServices.Models;

namespace CipherServices.Pages
{
  public class IndexModel : PageModel
  {
    public Dictionary<string, string> Secrets { get; set; }
    [BindProperty]
    public Message NewMessage { get; set; }

    private readonly IDecrypter _decrypter;
    private readonly IEncrypter _encrypter;
    private readonly MessageContext _context;

    public IndexModel(IDecrypter decrypter, IEncrypter encrypter, MessageC
    {
      _decrypter = decrypter;
      _encrypter = encrypter;
      _context = context;
    }
```

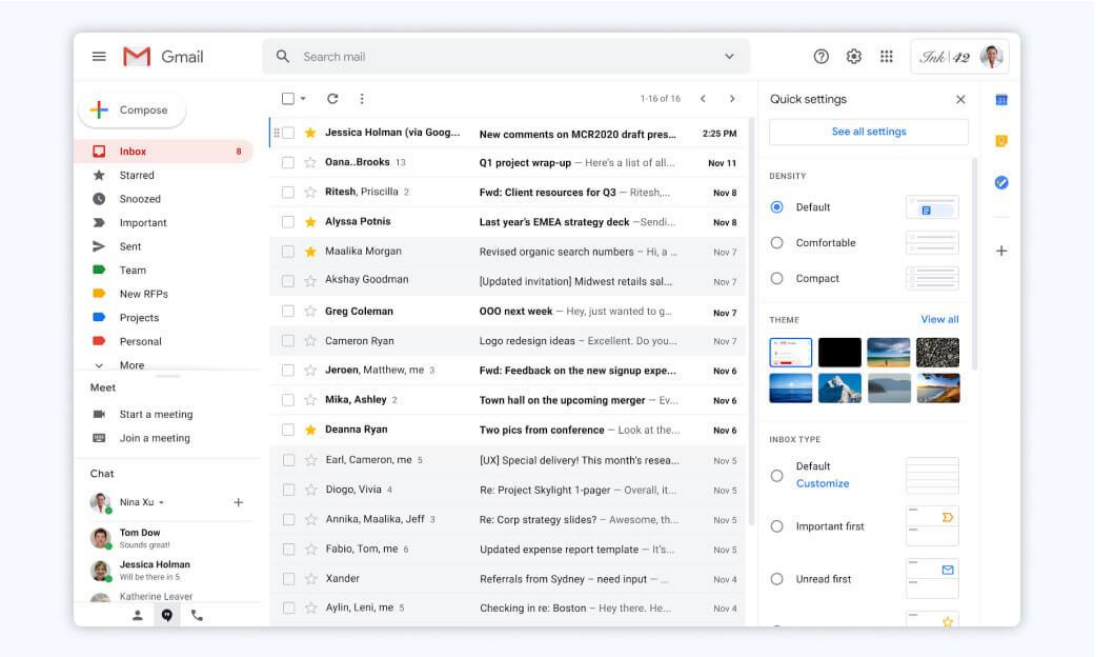
E-Mail

Email service

- One of the oldest network applications
- It is widely used at this moment
- There are two primary protocols:

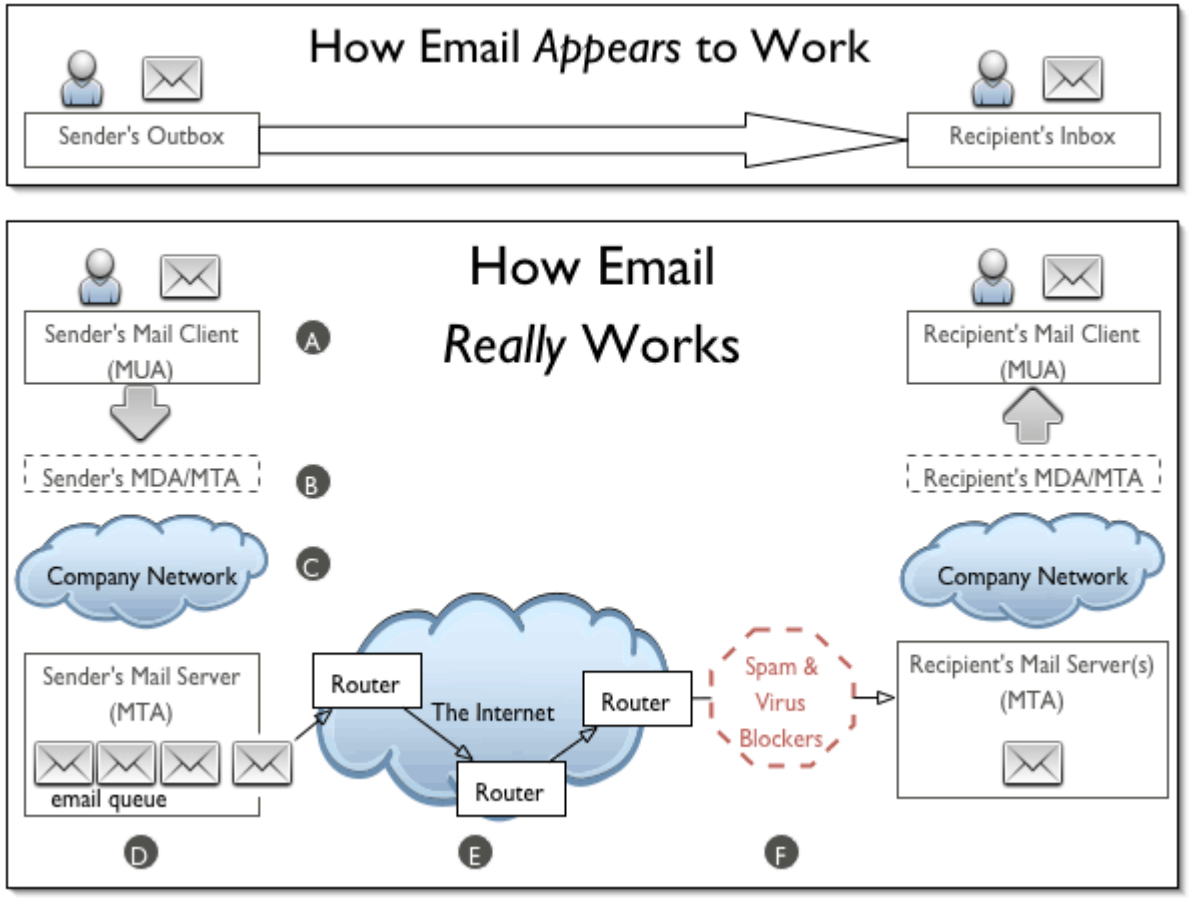
SMTP (outgoing), POP3/IMAP (incoming)

An example of Gmail



How an email system works

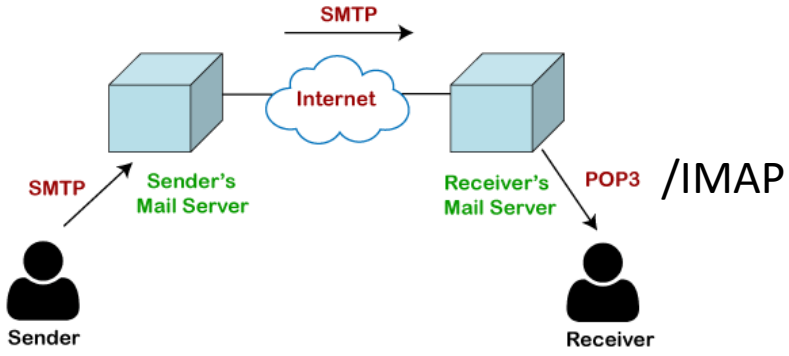
- Sender email: nvl@cs.ccu.edu.tw
- Receiver email: bob@gmail.com
- **Step A:** Sender creates and sends an email
- **Step B**: Sender's mail transfer agents (MTA) routes the email (e.g., using sendmail, postfix)
- **Step C:** Email is sent to the organization network 's mail server
- **Step D:** Email is ready to send at the mail server's email queue
- **Step E:** Email is sent out through Internet through intermediate MTA, they will find DNS of the recipient to forward
- **Step F:** An email may go through a spam/virus blocker (from ISPs or non-profit services).
- After that, email is transferred to the recipient's MTA server and the recipient's mailbox

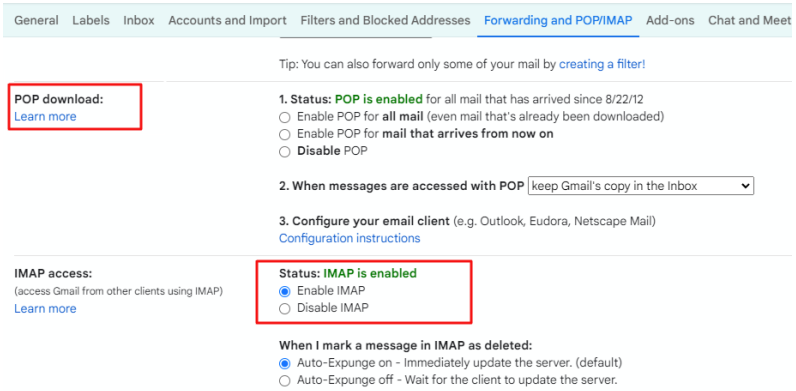


https://www.oasis-open.org/

POP3, IMAP, SMTP

POP3	IMAP	SMTP
Download all emails when there are Internet	Download emails on request	Sending out emails
Email accessing protocol (Mail reader)	Email accessing protocol (Mail reader)	Email transferring protocol (Mail sender)
Delete all emails on the server	Store all emails on the server	Doesn't store email on the server – just send out
Suitable for users with unstable Internet connection	Allow user to access from several devices	
Work at port 110	Work at port 143	Work at port 25, 443 (tls)

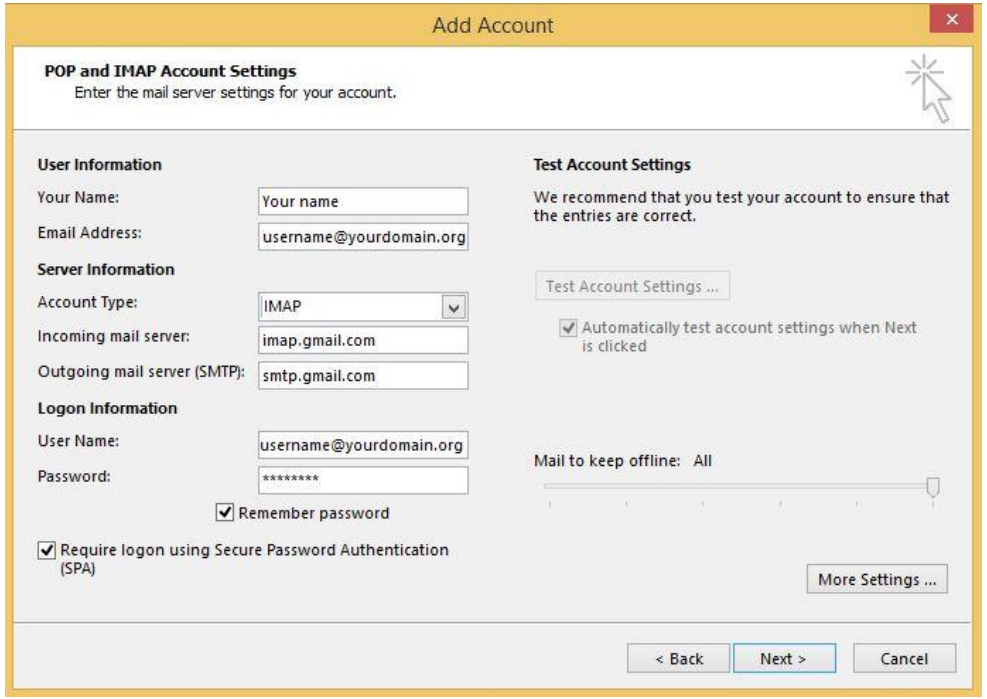


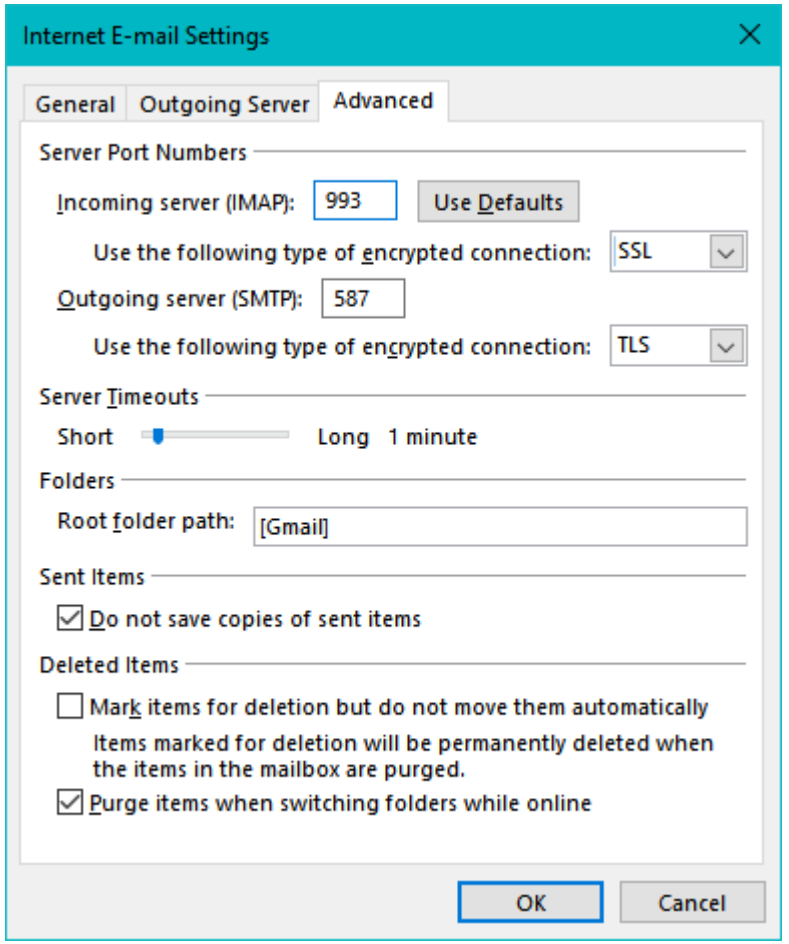


SMTP/IMAP setting

- You can set SMTP/IMAP

in your MS Outlook/Mailclient





When does your email have a problem?

- Your email is filtered and flagged by spam filter
- Sender/Receiver mail server's MTA email queue is full (disk storage)
- DNS server is not working
- Fail authentication for SMTP/IMAP services
- .....

Multimedia, Streaming services

Multimedia services

- **Popular apps**: Video call, voice over IP, and videoconferencing

- These apps use **Real-Time Transport Protocol** (RTP) that specifies the coding schemes and media types of an application

Suppose you want to hold a videoconference at a certain time and make it available to a wide number of participants.

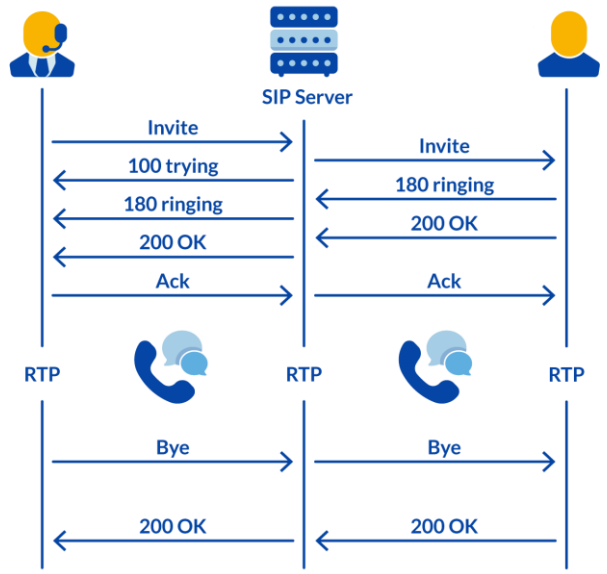
You have decided to encode the video stream using the MPEG-2 standard, to use the multicast IP address 224.1.1.1 for transmission of the data, and to send it using RTP over UDP port number 4000.

How would you make all that information available to the intended participants?

Session Initiation Protocol (SIP)

- The IETF has defined protocols for broadcasting information to the intended participants. The protocols that have been defined include
  - SDP (Session Description Protocol)
  - SAP (Session Announcement Protocol)
  - SIP (Session Initiation Protocol)
  - SCCP (Simple Conference Control Protocol)

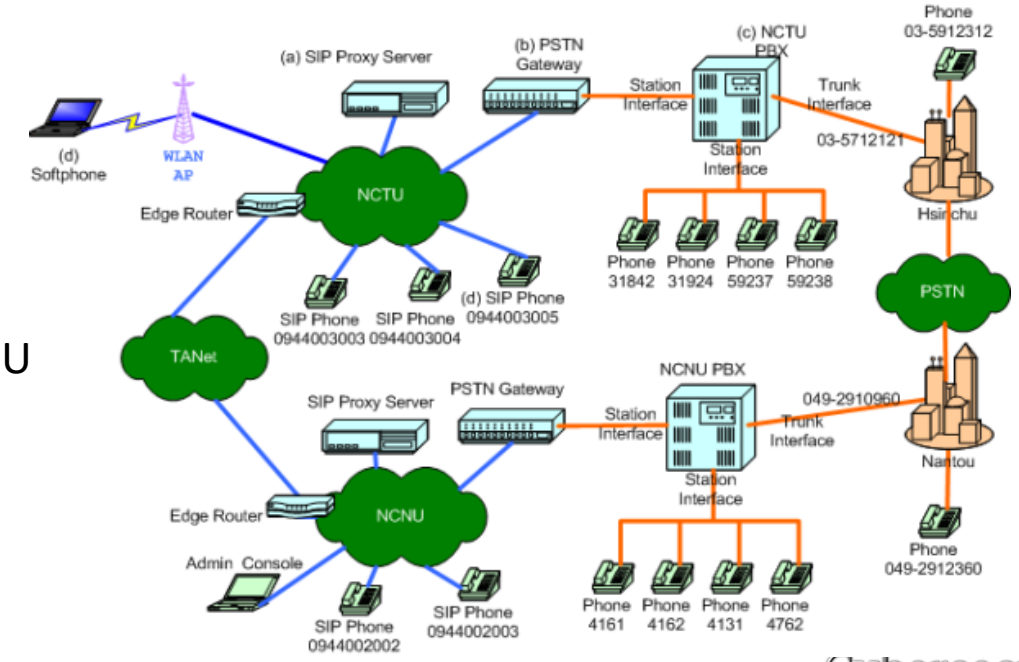
- SDP conveys the following information:
  - The name and purpose of the session
  - Start and end times for the session
  - The media types (e.g. audio, video) that comprise the session
  - Detailed information needed to receive the session (e.g. the multicast address to which data will be sent, the transport protocol to be used, the port numbers, the encoding scheme, etc.)



Session Initiation Protocol (SIP)

- SIP
  - SIP is an application layer protocol that bears a certain resemblance to HTTP, being based on a similar request/response model.
  - However, it is designed with rather different sorts of applications in mind, and thus provides quite different capabilities than HTTP.

An example of
VoIP service in NCTU/NCNU



DNS systems

Domain Name System

- **Goal**: Turns domain names into IP addresses, which allow browsers to get to websites and other internet resources

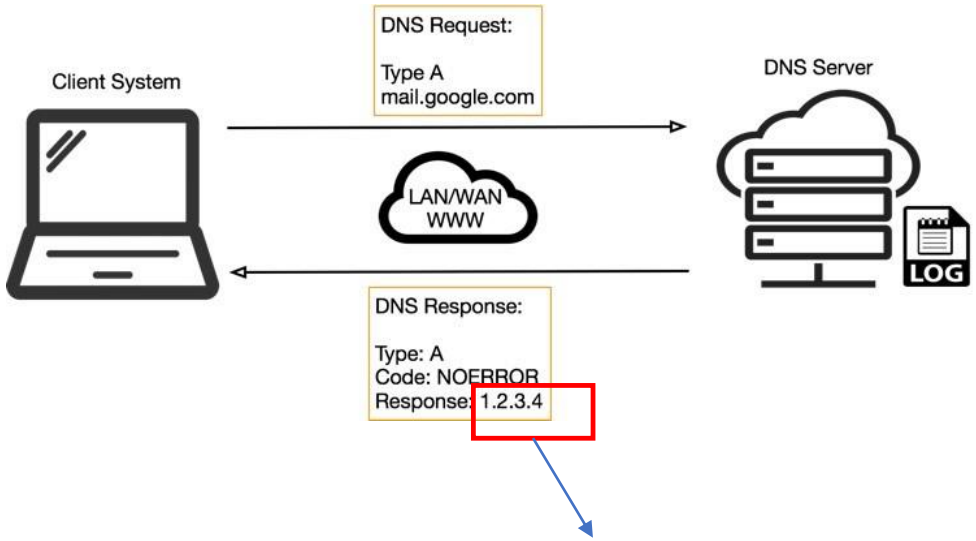
- Why using DNS? It is because it is difficult to remember IP address

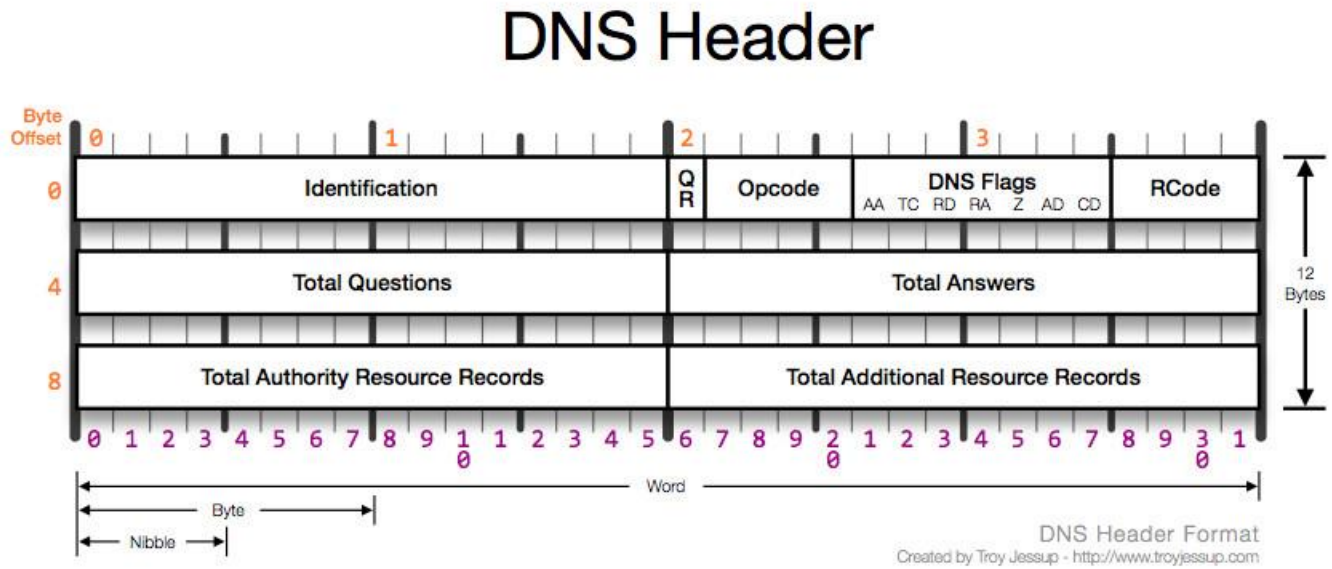
  ccu.edu.tw ← → 140.123.13.215:443

- DNS is a hierarchical and distributed naming system

Transfer domain name to IP address

Domain Nam System (DNS)

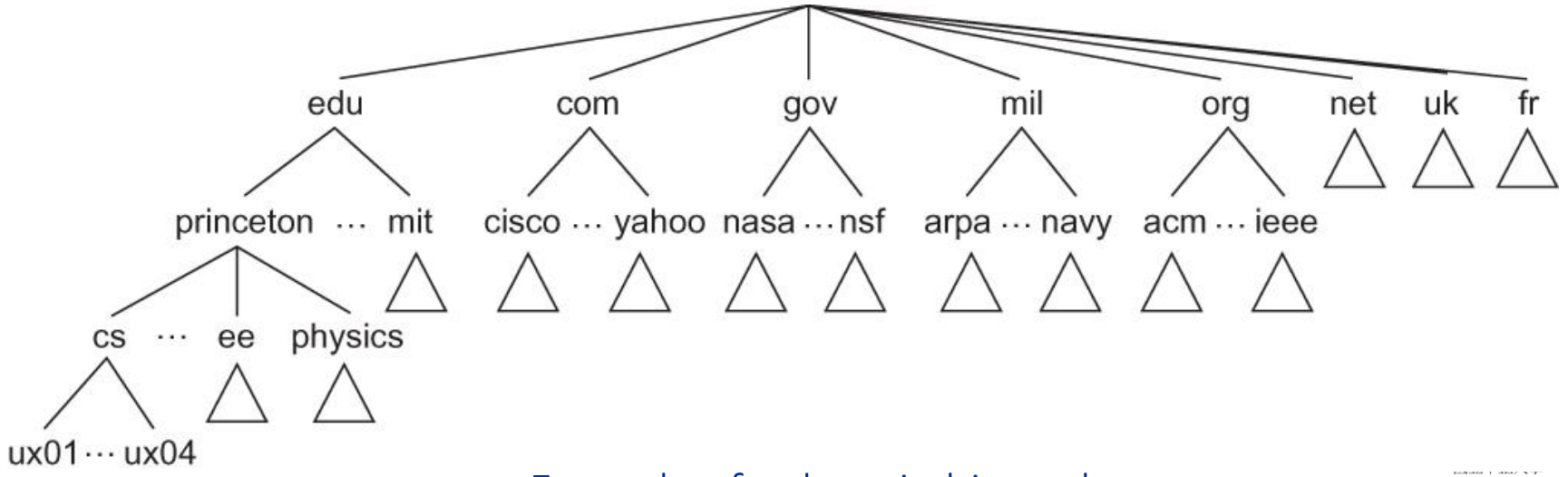




The DNS response includes IP address 1.2.3.4 of the domain name *mail.google.com*

DNS Hierarchy

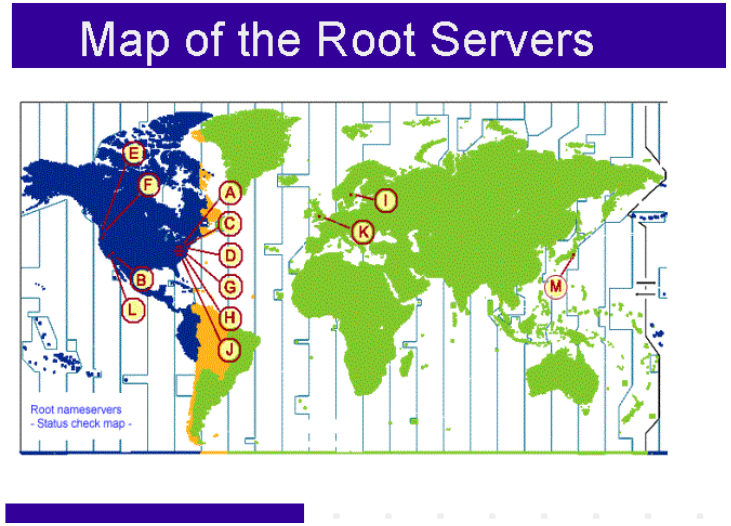
- Domain Hierarchy
  - DNS implements a hierarchical name space for Internet objects.
  - Like the Unix file hierarchy, the DNS hierarchy can be visualized as a tree, where each node in the tree corresponds to a domain, and the leaves in the tree correspond to the hosts being named.



Example of a domain hierarchy

DNS owner

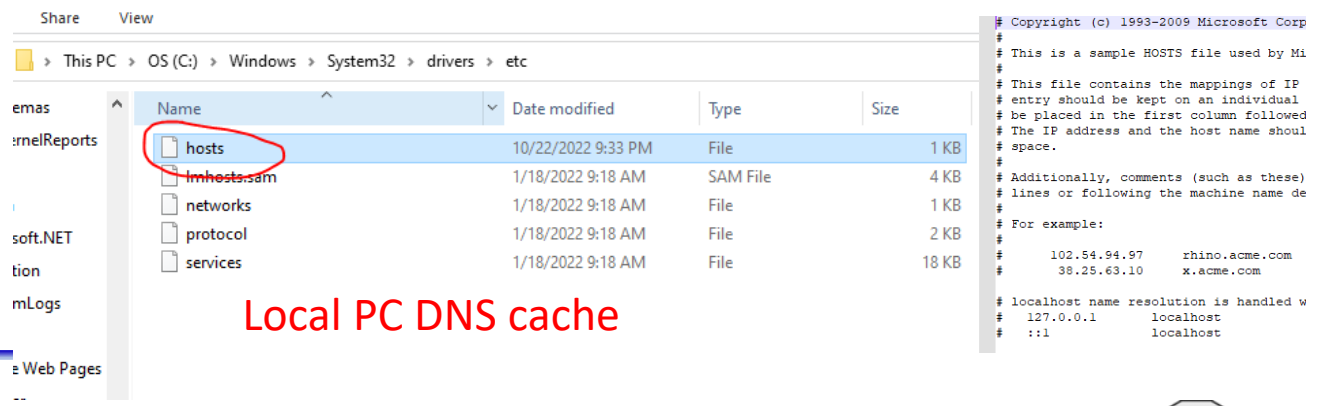
- Some administrative authority is responsible for that portion of the hierarchy.
- The top level of the hierarchy forms a zone that is managed by the Internet Corporation for Assigned Names and Numbers (ICANN).
- Root DNS servers are located distributely



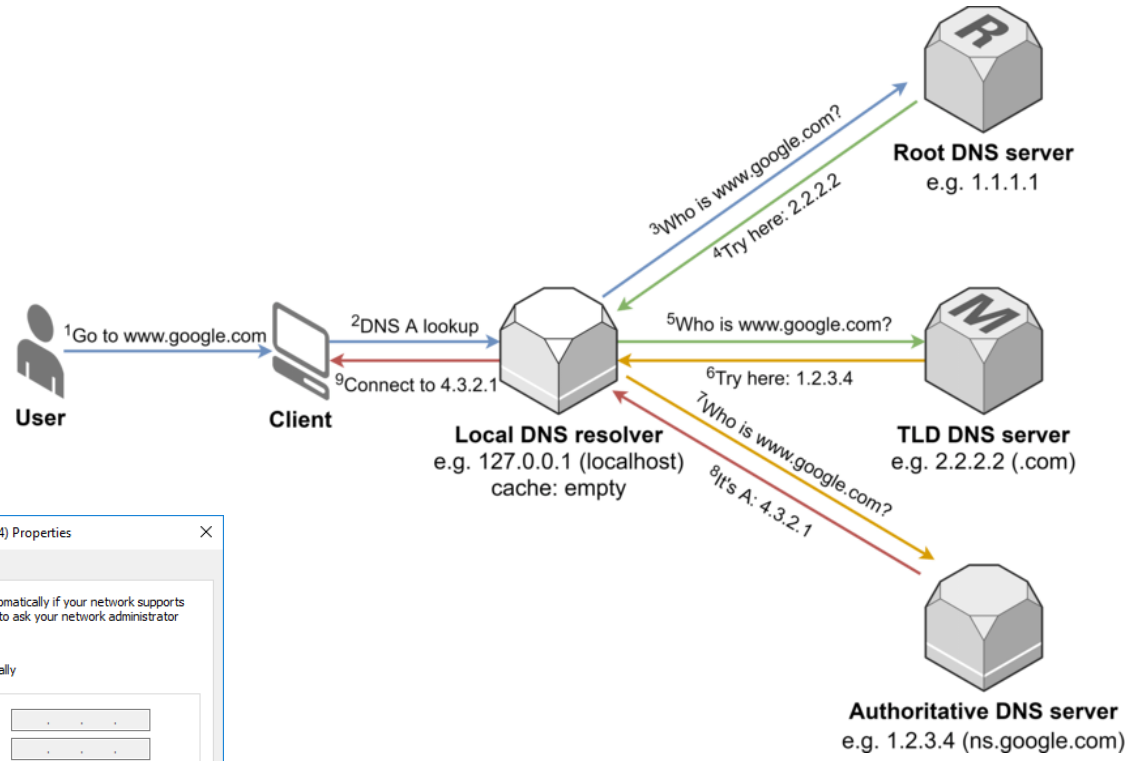
DNS resolution

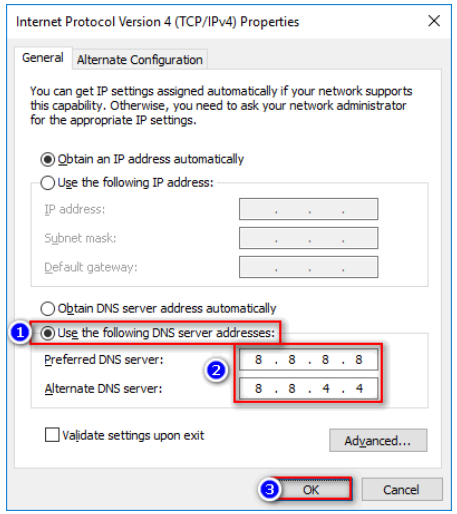
- An example of DNS resolution

- The order process
  1. Local PC DNS cache
  2. Your primary DNS domain setting
  3. Root DNS server
  4. Authoritative DNS server



Local PC DNS cache





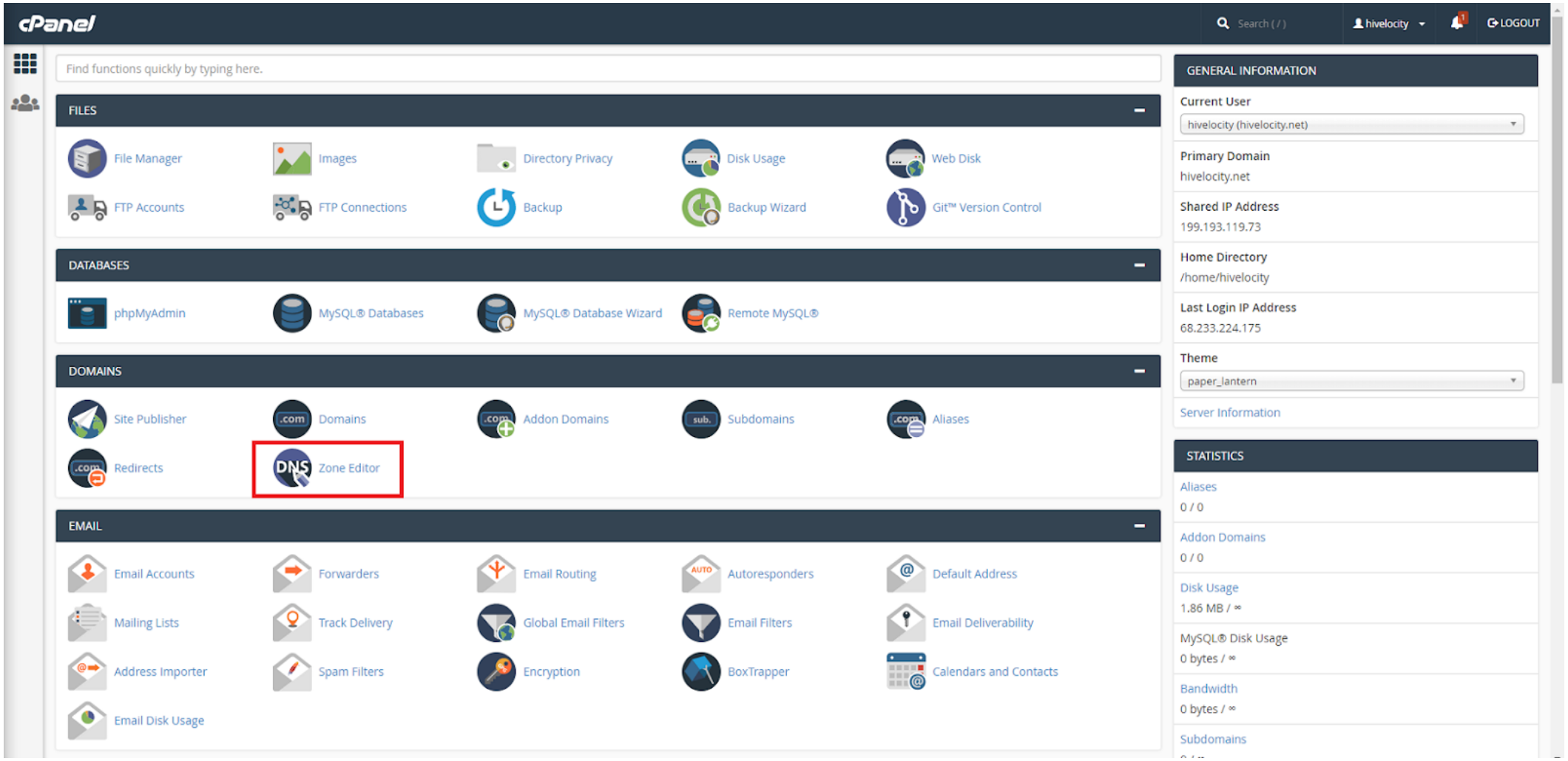
國立中正大學
Cybersecurity Lab

Name Servers

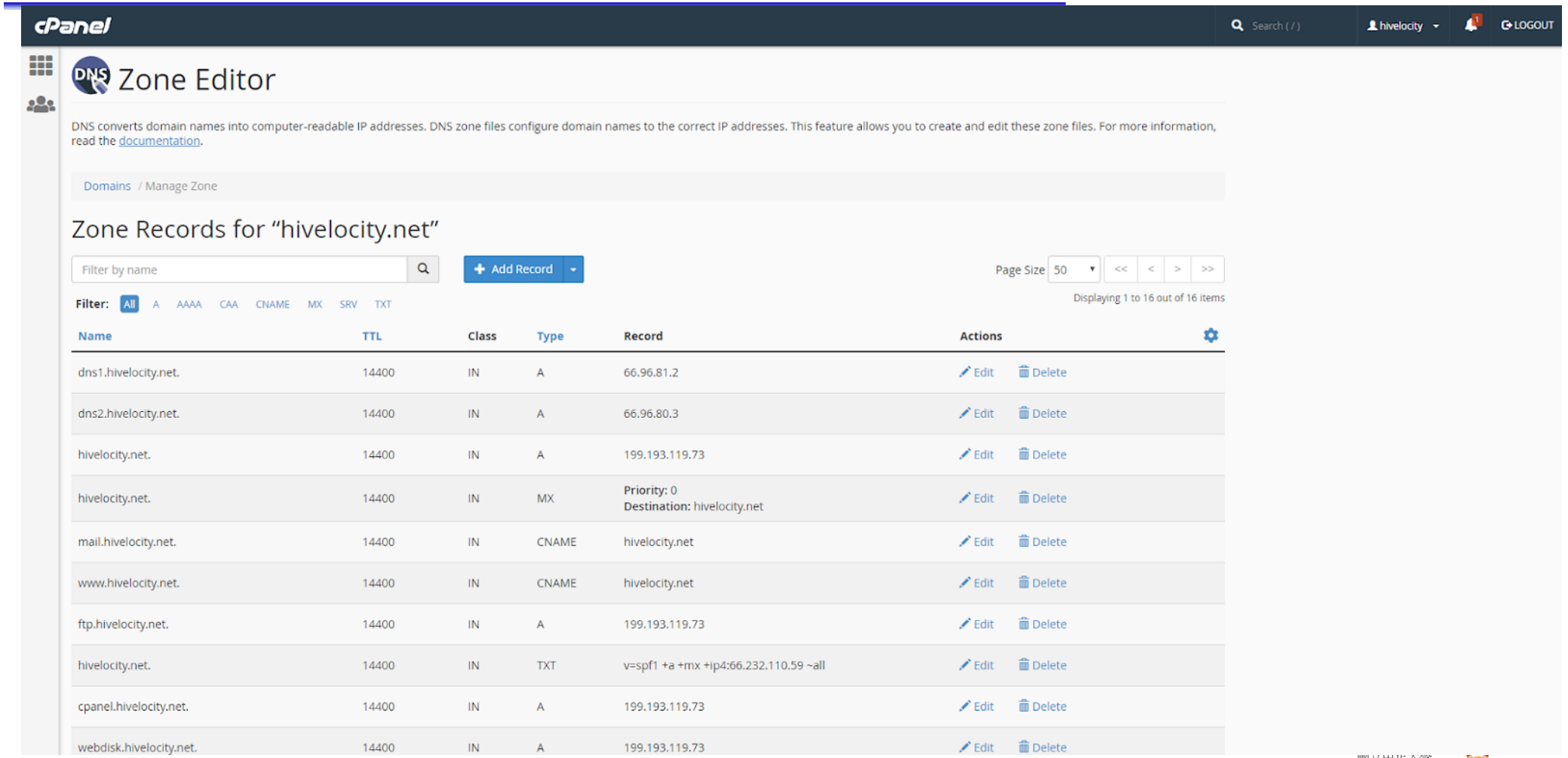
- Name Servers
  - Type = A indicates that the Value is an IP address. Thus, A records implement the name-to-address mapping we have been assuming.
  - NS: The Value field gives the domain name for a host that is running a name server that knows how to resolve names within the specified domain.
  - CNAME: The Value field gives the canonical name for a particular host; it is used to define aliases.
  - MX: The Value field gives the domain name for a host that is running a mail server that accepts messages for the specified domain.

Common DNS Record Types	
Record	**Description**
A	Address record (IPv4)
AAAA	Address record (IPv6)
CNAME	Canonical Name record
MX	Mail Exchanger record
NS	Nameserver record
PTR	Pointer record
SOA	Start of Authority record
SRV	Service Location record
TXT	Text record

DNS record in Cpanel



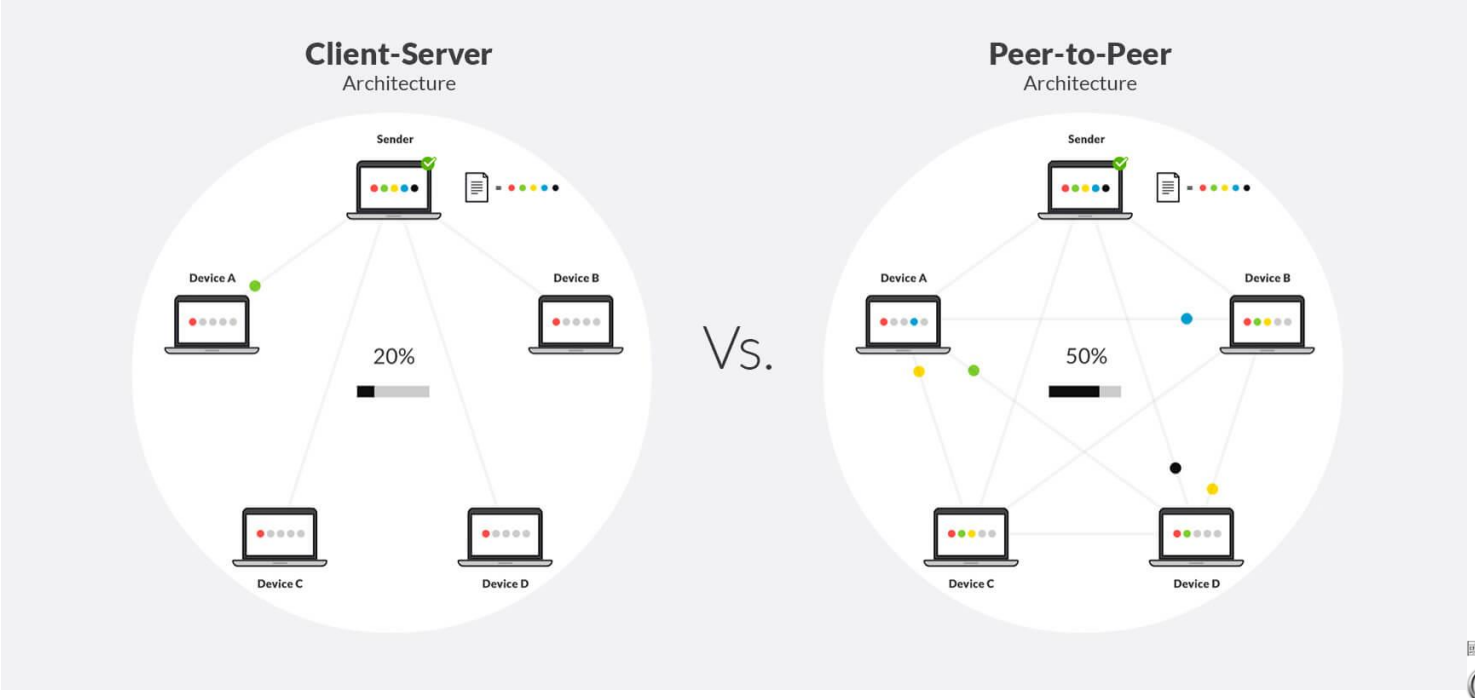
DNS record in Cpanel



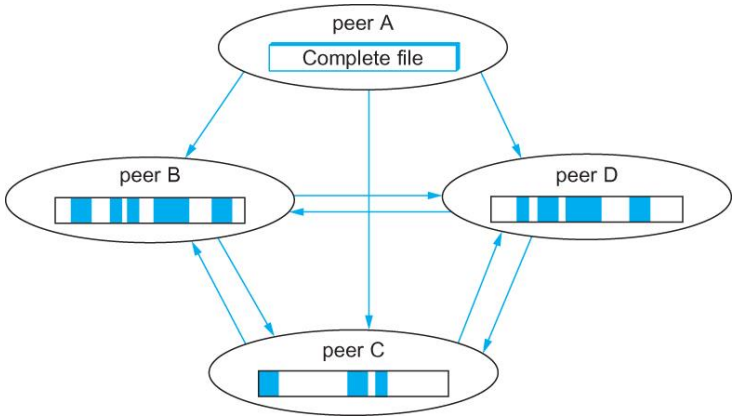
Popular network models

Peer-to-peer networks

- **Definition**: a distributed application architecture that partitions tasks or workloads between peers
- Exploit the power of thousands of nodes to download/share files without your having to contact a centralized authority → more freedom (the hacking code/pirate software sharing)



BitTorrent



- **Definition**: a communication protocol for peer-to-peer file sharing, which enables users to distribute data and electronic files over the Internet in a decentralized manner

- Each file is shared via its own independent BitTorrent network, called a *swarm*

- A node that wants to download the file joins the swarm, becoming its second member, and begins downloading pieces of the file from the original peer.

- In doing so, it becomes another source for the pieces it hasdownloaded, even if it has not yet downloaded the entire file

Peers in a BitTorrent swarm download from other peers that may not yet have the complete file