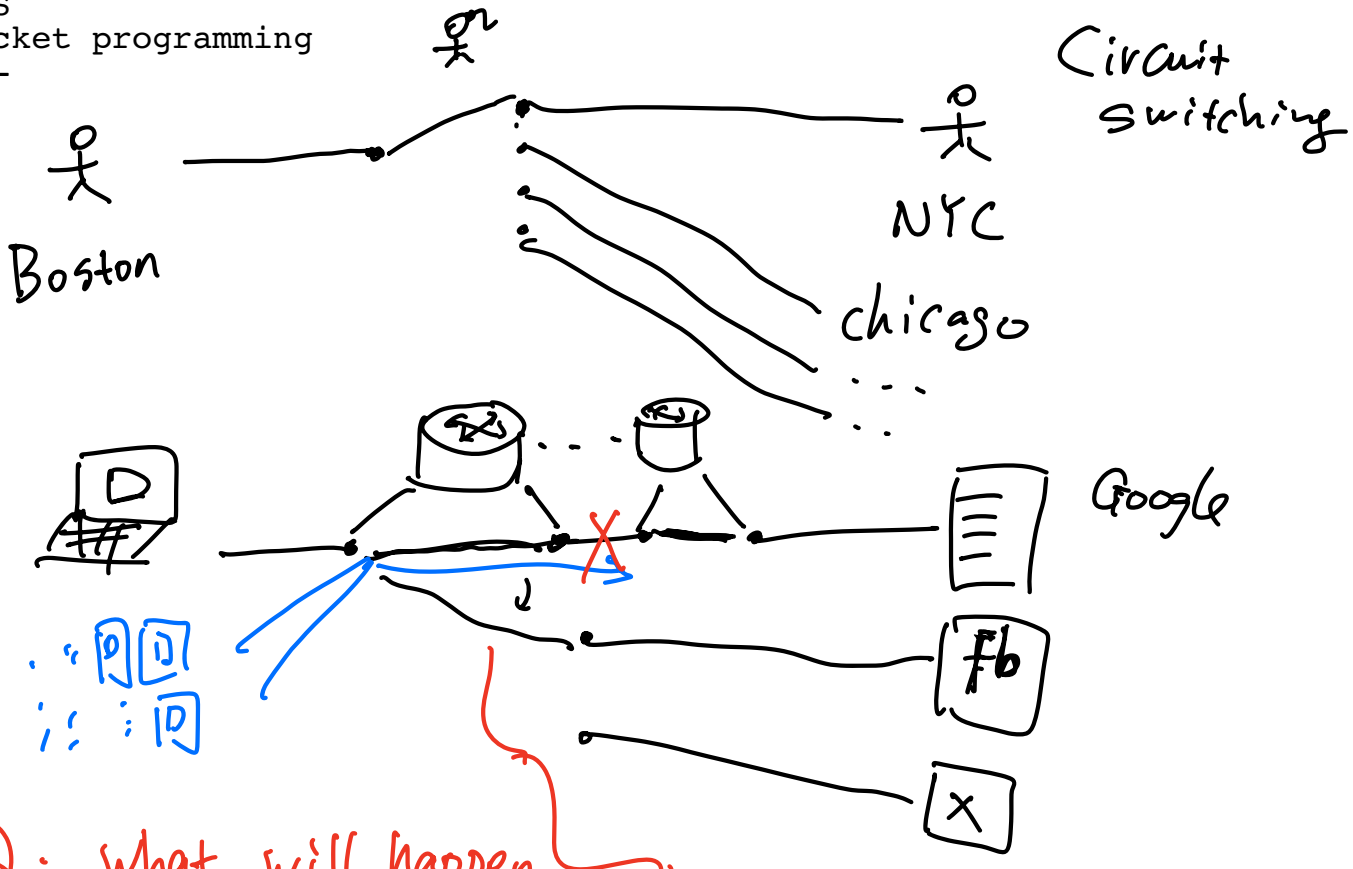
HOST NAMES

HOSTNAME	HOST ADDR (Dec)	LIAISON	STATUS
AFWL-TIP	176	D Hyde (505)247-1711 x3803	TIP, Up 3-74
ALOHA-TIP	164	R Binder (808)948-7066	TIP
AMES-11	208	J Hart (415)965-5935	USER, up 12-73
AMES-67	16	W Hathaway (415)965-6033	SERVER
AMES-TIP	144	W Hathaway (415)965-6033	TIP
ANL	?	L Amiot (312)739-7711 x4309	SERVER, up 2-74
ARPA-DMS	28	S Crocker (202)694-5037	USER, Agency use only
ARPA-TIP	156	S Crocker (202)694-5037	TIP
BBN-11X	5	R Thomas (617)491-1850 x483	Peripheral processor for #69, up 12-73
BBN-1D	232	A McKenzie (617)491-1850 x441	USER
BBN-NCC	40	A McKenzie (617)491-1850 x441	USER
BBN-TENEX	69	R Thomas (617)491-1850 x483	SERVER
BBN-TENEXB	133	R Thomas (617)491-1850 x483	SERVER, Limited
BBN-TESTIP	158	A McKenzie (617)491-1850 x441	TIP (magtape)
BELVOIR	27	W Andrews (703)664-5511	USER, up 6-74
BRL	29	M Romanelli (301)278-4574	USER
CASE-10	13	J Calvin (216)368-2984	SERVER
CCA-TENEX	31	R Winter (617)491-3670	SERVER
CCA-TIP	159	R Winter (617)491-3670	TIP
CMU-10A	78	H Van Zoeren (412)621-2600 x160	SERVER
CMU-10B	14	H Van Zoeren (412)621-2600,x160	SERVER
CMU-11	142	C Pierson (412)621-2600 x130	USER, up Spring 74
CMU-CC	206	D King (412)621-2600 x2683	USER, up Spring 74
DOCB-TIP	153	S Stevenson (303)499-1000 x3138	TIP
EGLIN	?	E Blackwell (904)882-3734	Up 3/74
ETAC	20	G Petregal (202)433-3911	USER, up Spring 74
ETAC-TIP	148	G Petregal (202)433-3911	TIP (magtape)
FNWC	33	M Reese (408)646-2817	USER, up 2-74
FNWC-TIP	161	M Reese (408)646-2817	TIP
GWC-TIP	152	A Wells(402)294-2968	TIP (magtape)
HARV-1	73	B Reussow (617)495-4147	USER
HARV-10	9	B Reussow (617)495-4147	SERVER
HARV-11	137	S Bradner (617)495-3864	USER
HASKINS	197	F Cooper (203)865-6163	USER (VDH), up Spring 74
HAWAII-500	100	J Davidson (808)944-7455	SERVER, up 1-74
HAWAII-ALOHA	36	R Binder (808)948-7066	USER, Up 12-73
I4-TENEX	15	J McConnell (408)735-0635	SERVER
I4-TENEXA	79	J McConnell (408)735-0635	Peripheral processor for #15
ILL-CAC	12	T Milke (217)333-8469	USER
ILL-NTS	76	T Milke (217)333-8469	USER
ISI-DEVTENEX	150	R Hoffman (213)822-1511 x190	USER, up 1-74
ISI-SPEECH11	22	R Hoffman (213)822-1511 x190	USER, up 1-74

Week 12.b
CS3650
03/27 2024
https://naizhengtan.github.io/24spring/

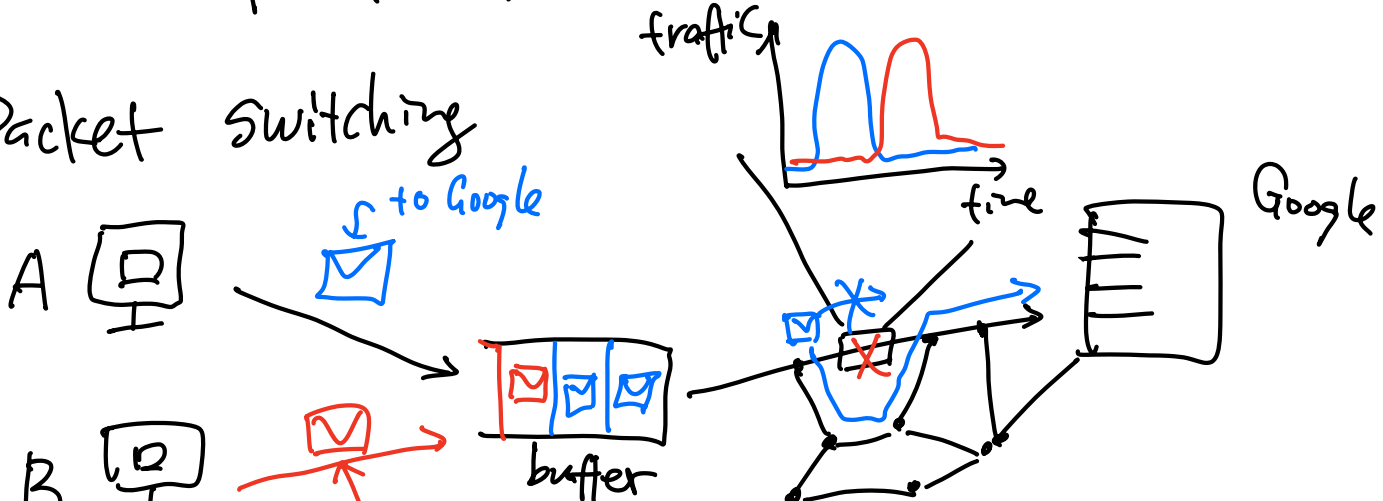
1. packet switching
2. IP
3. DNS
4. socket programming



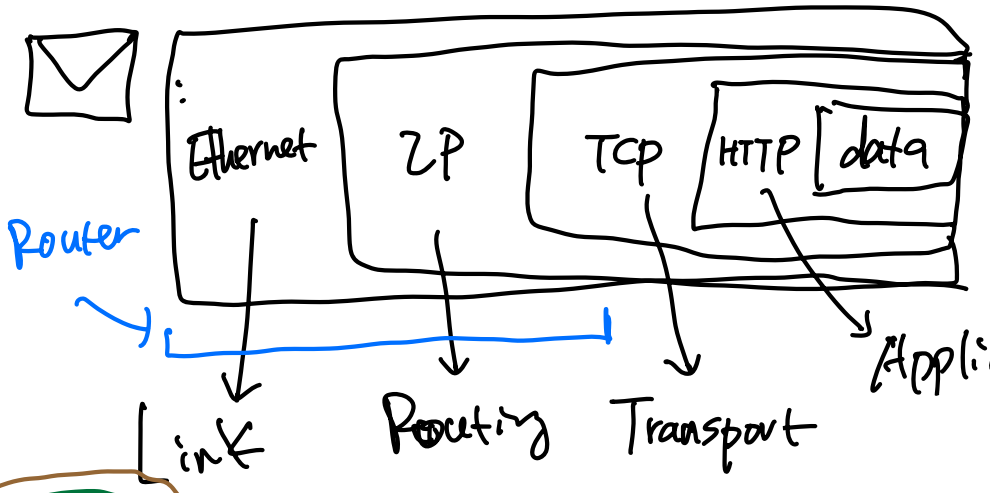
Q: what will happen for Internet?

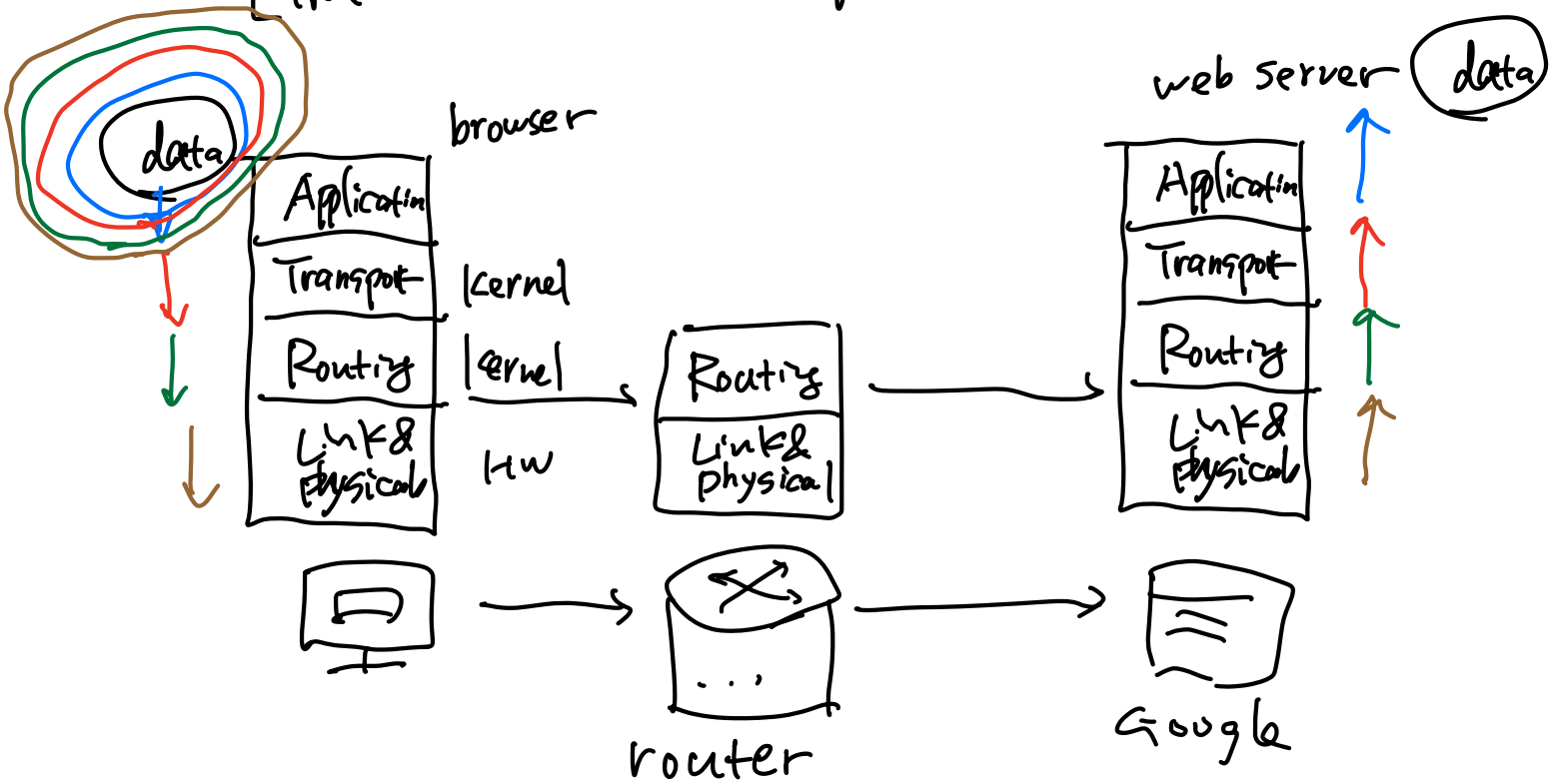
- overloading
  1. mostly idle
  2. setup a new connection each time visiting a page
  3. failure ⇒ disconnected

- Packet switching



to FB





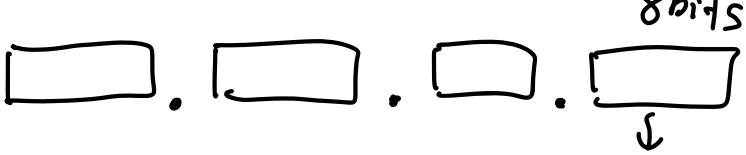
- IP addresses

Q: What does an IP look like?

192.123.123.1

IP: identifying hosts on Internet

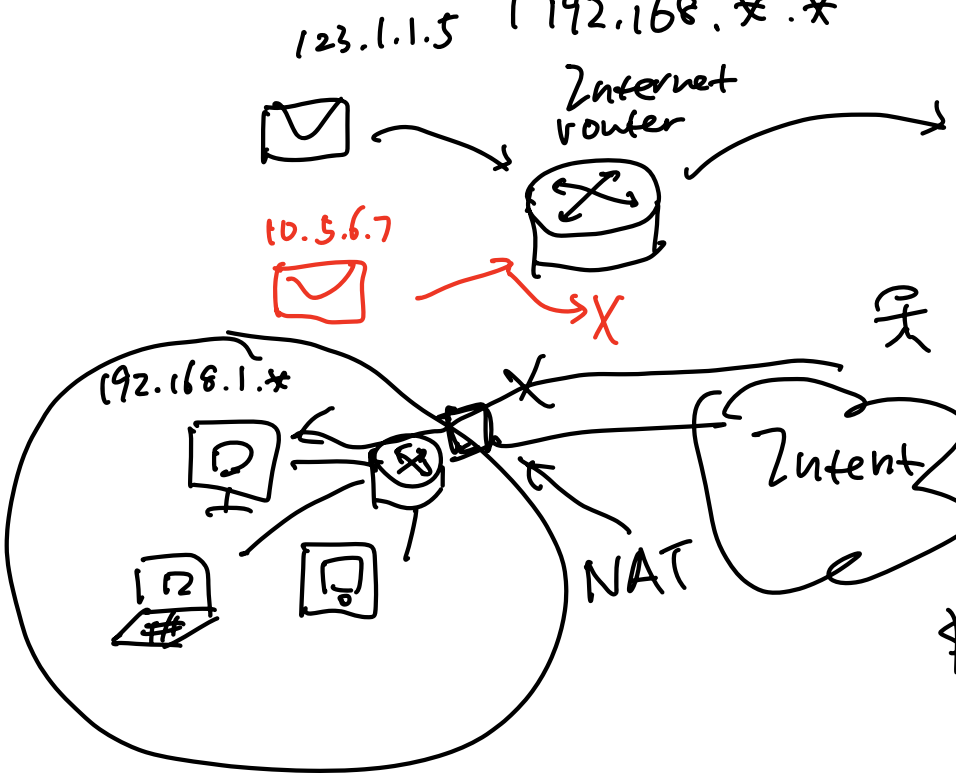
IP_{V4}: 32bits



$[0, \frac{2^8-1}{255}]$

0.0.0.0 — 255.255.255.255

- public IP : routing on public Internet
- private IP : 10.*.*.*, 172.16.*.* — 172.31.*.*, 192.168.*.*



$30/month
$50/month
$10/month

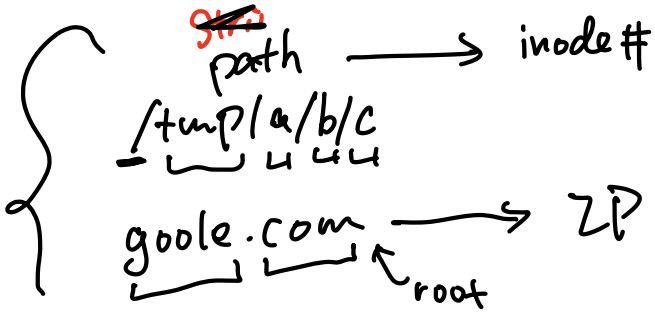
$3.6/month
$0.005/hr

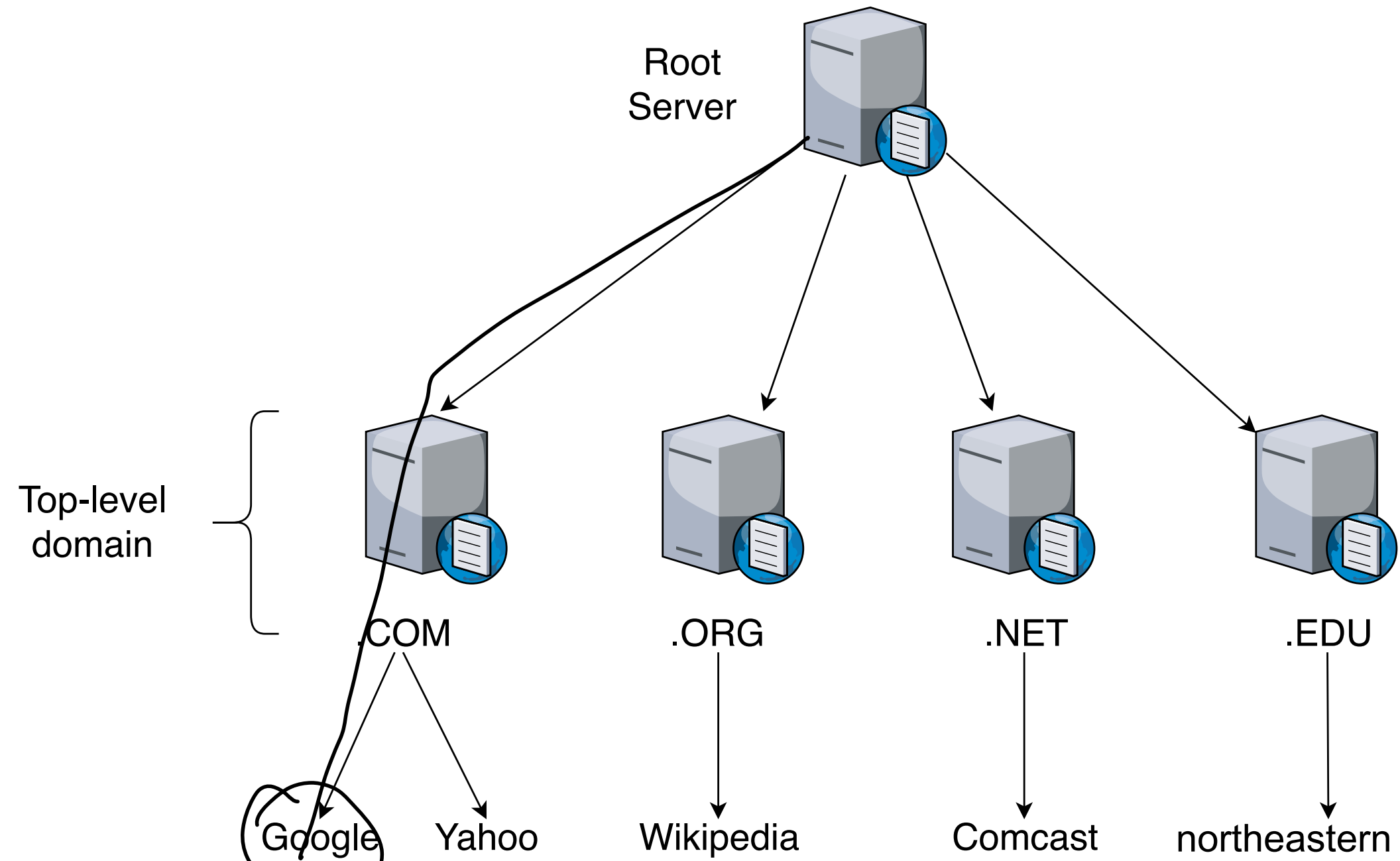
IANA

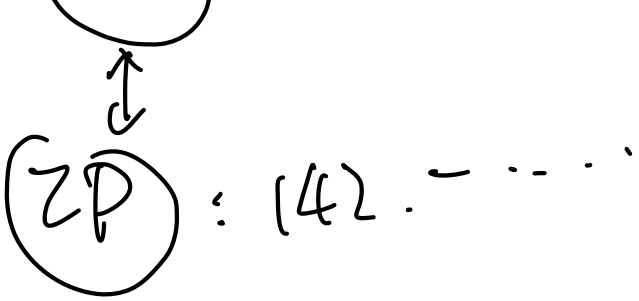
- DNS:
  Domain Name System

Q: We want a user-friendly way of remembering IP.

HOSTS.txt




Root
Server
Top-level
domain
.COM
.ORG
.NET
.EDU
Google
Yahoo
Wikipedia
Comcast
northeastern

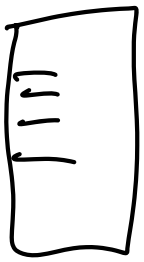


- Socket programming

Q: ip ↔ machine,

port number, 16-bit

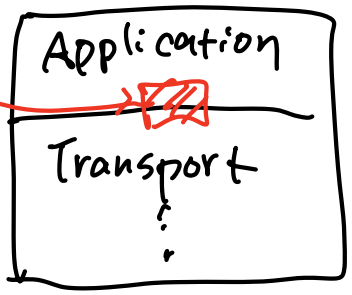
<1024



HTTP

web server (80)

ssh (22)



```
CS3650: socket programming

// 1. This is a simple example of a client sending "hello world!"
//    to a server.

     [server]                    [client]
        |                           |
     fd = socket(...)            fd = socket(...)
     bind(fd,...)                   |
     listen(fd,...)                 |
        |                           |
  new_fd = accept(fd,...)      +--connect(fd,...)
        |                    /      |
        |<-------------------+       |
        |--------------------------->|
        |                            |
     new_fd <====================== fd


// 2. Server code

  // assuming the following helper function will fill in the "struct sockaddr"
  void init_sockaddr(struct sockaddr *in_addr, const char *ip, int port);

  // return a file descriptor
  int listen_socket() {
      int fd = socket(AF_INET, SOCK_STREAM, 0);

      struct sockaddr addr;
      init_sockaddr(&addr, NULL, 3650 /*port number*/);

      bind(fd, &addr, sizeof(addr));

      listen(fd, 128);

      struct sockaddr tmp;
      socklen_t addr_size = sizeof(tmp);
      int new_fd = accept(fd, &tmp, &addr_size);

      close(fd); // stop accepting more connections

      return new_fd;
  }

  int main() {
      int new_fd = listen_socket();

      char buf[1024] = {0};
      recv(new_fd, &buf, 1024, 0); // receiving data
      printf("%s\n", buf);

      close(new_fd);
  }
```



```

// 3. Client code

int connect_socket() {
    int fd = socket(AF_INET, SOCK_STREAM, 0);

    struct sockaddr serv_addr;
    init_sockaddr(&serv_addr, "127.0.0.1" /* ip */, 3650 /* port */);

    connect(fd, &serv_addr, sizeof(serv_addr));

    return fd;
}

int main() {
    int fd = connect_socket();

    char *hello = "hello world!";
    send(fd, hello, strlen(hello), 0); // sending data

    close(fd);
}
```

4. Socket programming interfaces:

a) socket, send, and recv

* int **socket**(int domain, int type, int protocol);

  socket() creates an endpoint for communication and returns a descriptor.

* ssize_t **send**(int socket, const void *buffer, size_t length, int flags);

  send a message from a socket

* ssize_t **recv**(int socket, void *buffer, size_t length, int flags);

  receive a message from a socket

b) bind, listen, and accept (server side)

* int **bind**(int socket, const struct sockaddr *address, socklen_t address_len);

  bind() assigns a name to an unnamed socket.

* int **listen**(int socket, int backlog);

  listen for connections on a socket

* int **accept**(int socket, struct sockaddr *restrict address,
            socklen_t *restrict address_len);

  accept a connection on a socket

c) connect (client side)

* int **connect**(int socket, const struct sockaddr *address, socklen_t address_len);

  initiate a connection on a socket