Application Layer

- processes communicate NOT programs
  → process: programs in execution.

- Interprocess comm^tn from CS 314
  - Files
  - Signals
  - Pipes
  - Message queues
  - Semaphores
  - Shared memory
  - Message passing.

↑ these are processes running on the same end system.

Network Communication = Interprocess Comm^tn + across a network.

Processes running on different end systems.

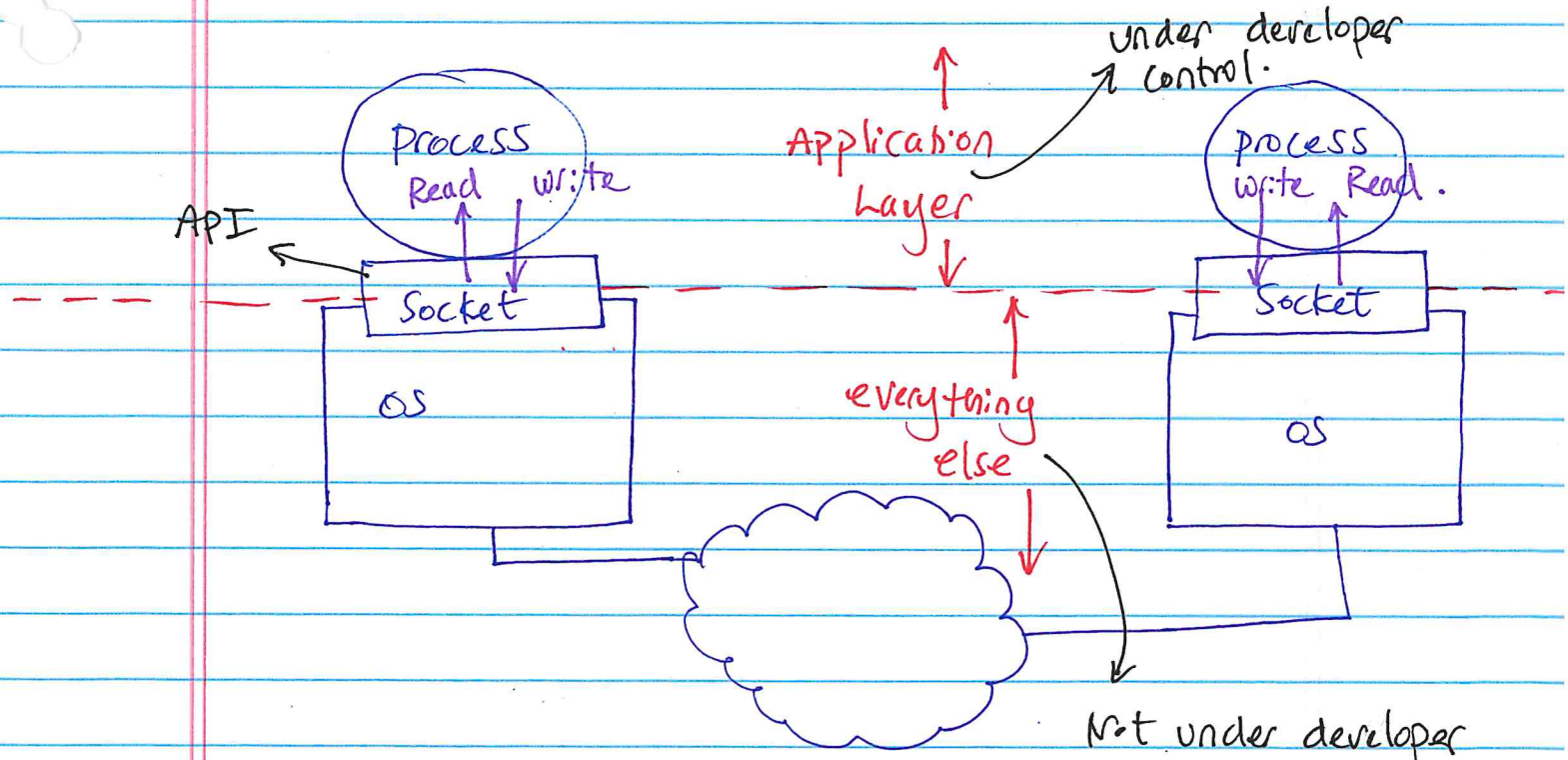
- Two main Comm^tn models

Client-Server
- Client: Initiates comm^tn
- Server: waits and respond to client requests

Peer-to-peer (P2P)
- each peer acts as a client AND a server

- The communication primitive used is called a "socket".

Socket : An API to interact with the OS (which implements the other layers)
- used to send and receive 'messages' from other processes.
- Other alternatives : STREAM, Transport Layer Interface (TLI)
- No memory. Merely a data structure.
  ↳ Door Analogy



- Sockets provide the "Interface" to what's not under developer's control.

- Sockets have a logical end-to-end conn^tn with each other

- Sockets have unique addresses
  - NL address : To locate end system or node
  - TL address : to locate the correct process.

Socket Address ≡ Network Address : port number (IP address)

e.g. IPV4

192.168.242.112 : 80

32-bits — 192.168.242.112

16-bits — 80

192.168.242.112 → identify the end system

80 → port.

TCP sockets

netstat -vatn

ss -t -a

UDP sockets

netstat -vaun

ss -u -a

IPV6

[ffff:0:a88:63e5:0243:85a3] : 80

128-bit (in hexa) — ffff:0:a88:63e5:0243:85a3

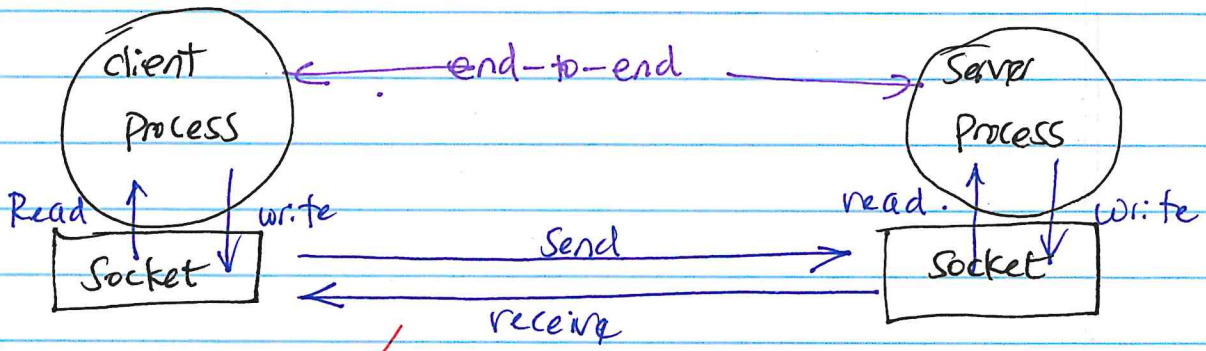
[ffff:0:a88:63e5:0243:85a3] → identify the end system

80 → port

Port # field

- 16-bits - 0-65535 ports
  - 0-1023 reserved
  - 1024-49151 Registered
  - 49152-65535 Dynamic / ephemeral.

Application Developer / programmer's perspective



NOTE: A logical end-to-end connection.

Each party need to locate each other's socket.

Client side

- local port: Randomly assigned @ time of connection.
- local IP: provided by OS.

Remote port and Remote IP
- Server must advertise

Server Side

- local port: specifically requested
  - bind().
- local IP: OS provided

Remote port & IP
- On client request

recall: Client initiates connection.

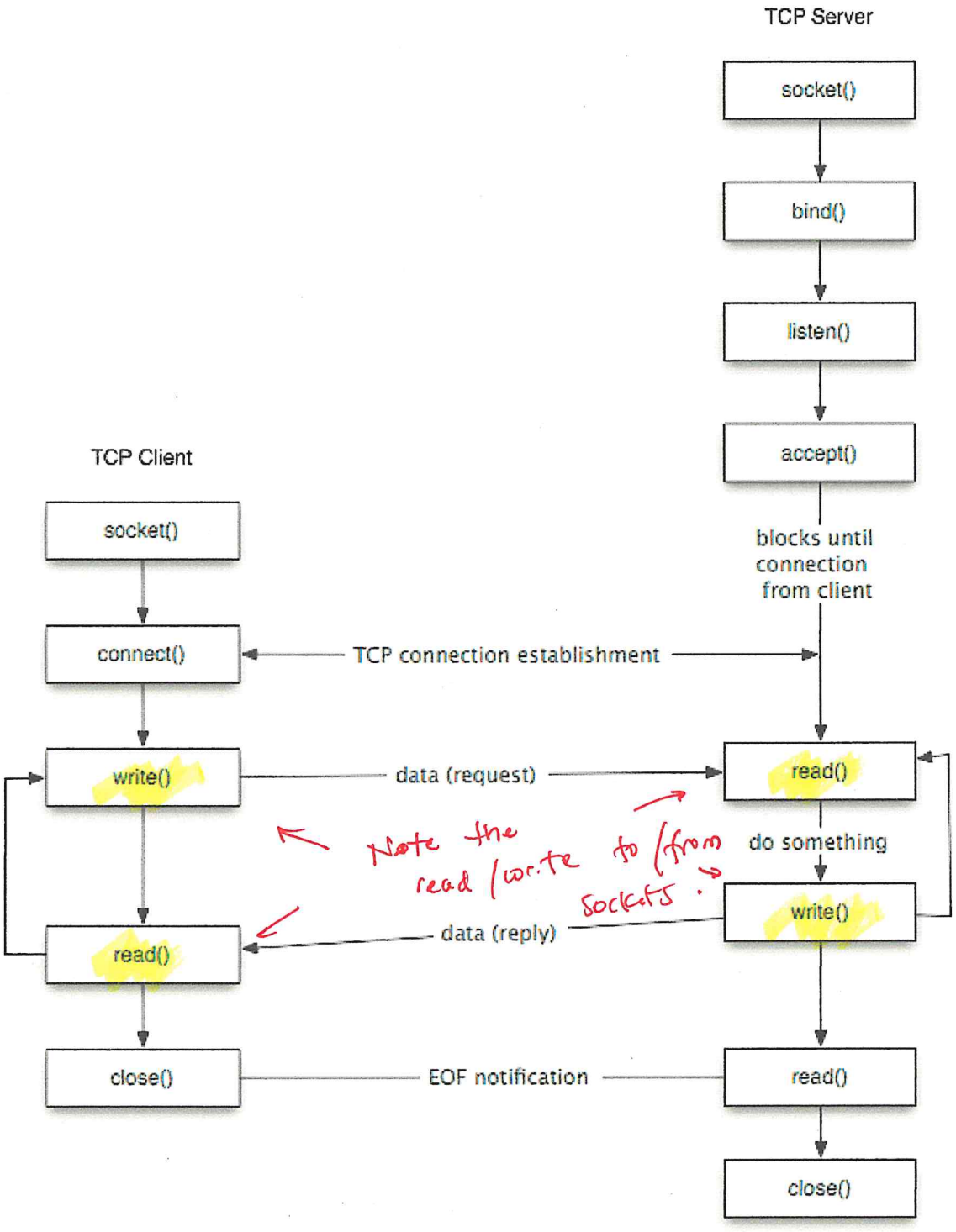


Figure 3: TCP client-server.

As shown in the figure, the steps for establishing a TCP socket on the client side are the following:

- Create a socket using the `socket()` function;
- Connect the socket to the address of the server using the `connect()` function;
- Send and receive data by means of the `read()` and `write()` functions.
- Close the connection by means of the `close()` function.

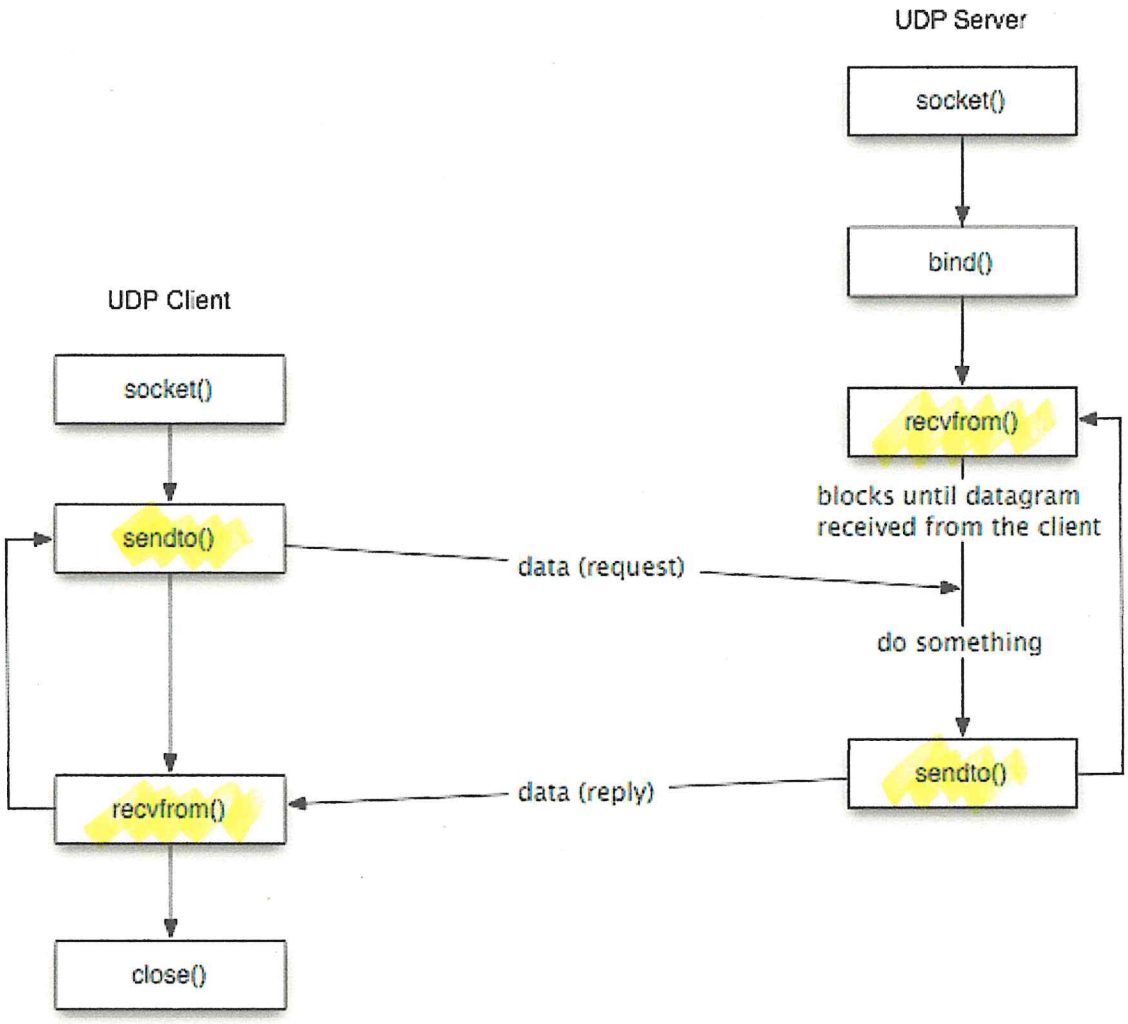


Figure 4: UDP client-server.

In this section, we will describe the two new functions `recvfrom()` and `sendto()`.

The recvfrom() Function

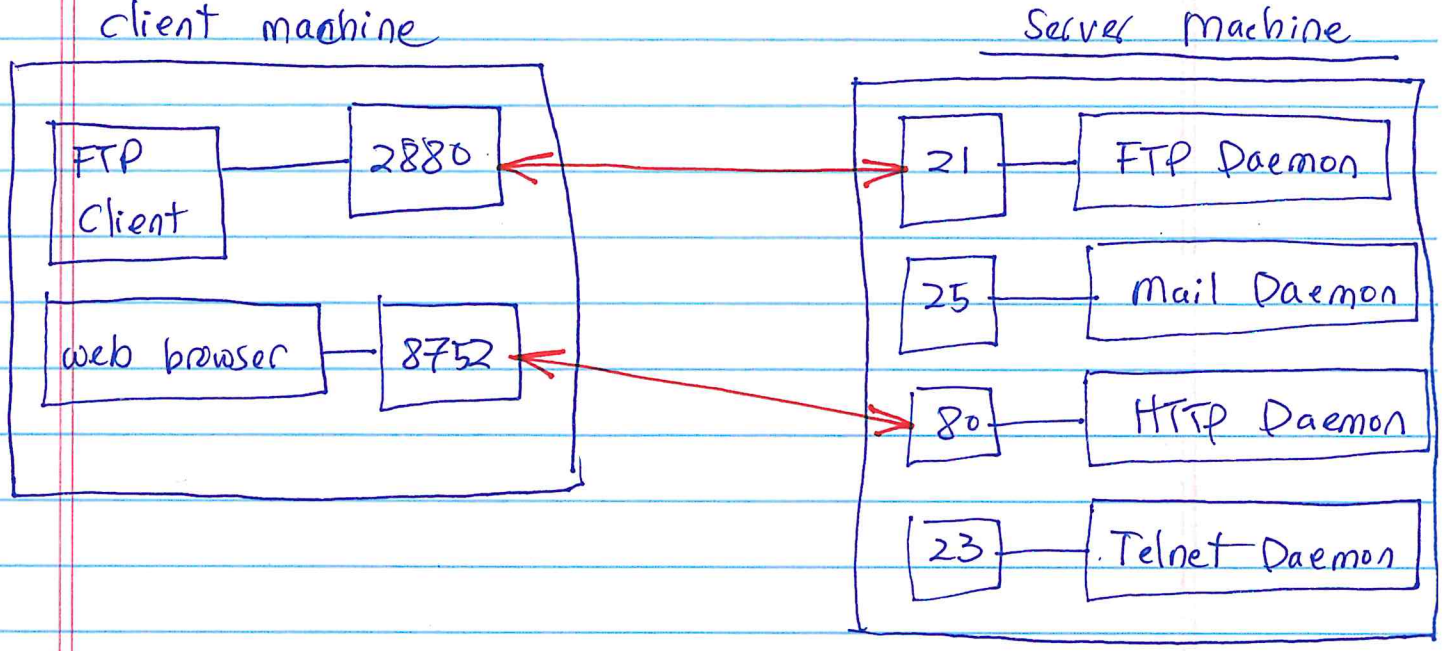
This function is similar to the `read()` function, but three additional arguments are required. The `recvfrom()` function is defined as follows:

```
#include <sys/socket.h>

ssize_t recvfrom(int sockfd, void* buff, size_t nbytes,
                          int flags, struct sockaddr* from,
                          socklen_t *addrlen);
```

The first three arguments `sockfd`, `buff`, and `nbytes`, are identical to the first three arguments of `read` and `write`. `sockfd` is the socket descriptor, `buff` is the pointer to read into, and `nbytes` is number of bytes to read. In our examples we will set all the values of the `flags` argument to 0. The `recvfrom` function fills in the socket address structure pointed to by `from` with the protocol address of who sent the datagram. The

- Clients as well as servers can have multiple socket/port combos open for different protocols.



- Servers for known protocols run on known port numbers [0 - 1023]
  - e.g. HTTP port 80.

- Clients run on arbitrary port numbers.

Common Ports

packetlife.net

TCP/UDP Port Numbers

7	Echo	554	RTSP	2745	Bagle.H	6891-6901	Windows Live
19	Chargen	546-547	DHCPv6	2967	Symantec AV	6970	Quicktime
20-21	FTP	560	rmonitor	3050	Interbase DB	7212	GhostSurf
22	SSH/SCP	563	NNTP over SSL	3074	XBOX Live	7648-7649	CU-SeeMe
23	Telnet	587	SMTP	3124	HTTP Proxy	8000	Internet Radio
25	SMTP	591	FileMaker	3127	MyDoom	8080	HTTP Proxy
42	WINS Replication	593	Microsoft DCOM	3128	HTTP Proxy	8086-8087	Kaspersky AV
43	WHOIS	631	Internet Printing	3222	GLBP	8118	Privoxy
49	TACACS	636	LDAP over SSL	3260	iSCSI Target	8200	VMware Server
53	DNS	639	MSDP (PIM)	3306	MySQL	8500	Adobe ColdFusion
67-68	DHCP/BOOTP	646	LDP (MPLS)	3389	Terminal Server	8767	TeamSpeak
69	TFTP	691	MS Exchange	3689	iTunes	8866	Bagle.B
70	Gopher	860	iSCSI	3690	Subversion	9100	HP JetDirect
79	Finger	873	rsync	3724	World of Warcraft	9101-9103	Bacula
80	HTTP	902	VMware Server	3784-3785	Ventrilo	9119	MXit
88	Kerberos	989-990	FTP over SSL	4333	mSQL	9800	WebDAV
102	MS Exchange	993	IMAP4 over SSL	4444	Blaster	9898	Dabber
110	POP3	995	POP3 over SSL	4664	Google Desktop	9988	Rbot/Spybot
113	Ident	1025	Microsoft RPC	4672	eMule	9999	Urchin
119	NNTP (Usenet)	1026-1029	Windows Messenger	4899	Radmin	10000	Webmin
123	NTP	1080	SOCKS Proxy	5000	UPnP	10000	BackupExec
135	Microsoft RPC	1080	MyDoom	5001	Slingbox	10113-10116	NetIQ
137-139	NetBIOS	1194	OpenVPN	5001	iperf	11371	OpenPGP
143	IMAP4	1214	Kazaa	5004-5005	RTP	12035-12036	Second Life
161-162	SNMP	1241	Nessus	5050	Yahoo! Messenger	12345	NetBus
177	XDMCP	1311	Dell OpenManage	5060	SIP	13720-13721	NetBackup
179	BGP	1337	WASTE	5190	AIM/ICQ	14567	Battlefield
201	AppleTalk	1433-1434	Microsoft SQL	5222-5223	XMPP/Jabber	15118	Dipnet/Oddbob
264	BGMP	1512	WINS	5432	PostgreSQL	19226	AdminSecure
318	TSP	1589	Cisco VQP	5500	VNC Server	19638	Ensim
381-383	HP Openview	1701	L2TP	5554	Sasser	20000	Usermin
389	LDAP	1723	MS PPTP	5631-5632	pcAnywhere	24800	Synergy
411-412	Direct Connect	1725	Steam	5800	VNC over HTTP	25999	Xfire
443	HTTP over SSL	1741	CiscoWorks 2000	5900+	VNC Server	27015	Half-Life
445	Microsoft DS	1755	MS Media Server	6000-6001	X11	27374	Sub7
464	Kerberos	1812-1813	RADIUS	6112	Battle.net	28960	Call of Duty
465	SMTP over SSL	1863	MSN	6129	DameWare	31337	Back Orifice
497	Retrospect	1985	Cisco HSRP	6257	WinMX	33434+	traceroute
500	ISAKMP	2000	Cisco SCCP	6346-6347	Gnutella		
512	rexec	2002	Cisco ACS	6500	GameSpy Arcade		
513	rlogin	2049	NFS	6566	SANE		
514	syslog	2082-2083	cPanel	6588	AnalogX		
515	LPD/LPR	2100	Oracle XDB	6665-6669	IRC		
520	RIP	2222	DirectAdmin	6679/6697	IRC over SSL		
521	RIPng (IPv6)	2302	Halo	6699	Napster		
540	UUCP	2483-2484	Oracle DB	6881-6999	BitTorrent		

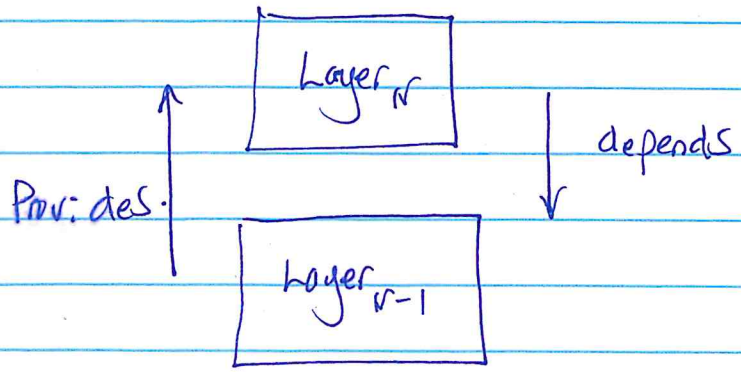
Legend

- Chat
- Encrypted
- Gaming
- Malicious
- Peer to Peer
- Streaming

IANA port assignments published at **http://www.iana.org/assignments/port-numbers**

by Jeremy Stretch

Service Level Agreement



- Since AL is at the top of the stack, NOT providing services to anyone.

- what are some desirable services from TL ?.
  - reliable data transfer
  - guarantees on throughput
  - Timing (QoS)
  - Security, Integrity.
  - Reliability

enumerate this list with students. Discuss potential applications

Discussion points

- Reliable Data transfer.
  - packets get lost, delayed, corrupt, arrive out of sync
    - discuss/ask students why these things could happen.
  - not all applications can tolerate such issues.
    - e.g. File transfer, email, etc.
  - AR must ensure reliable data transfer
    - A end-to-end service

- Guranteed performance
  - Some applications require bounded throughput
    - e.g. Internet Telephony (VoIP) require 32 kbps rate.
  - Some applications have strong timing requirements
    - e.g. real-time applications need data delivered within a bounded time
  - Applications that are bandwidth-sensitive vs. Elastic

- Security
  - Ensure only the intended end point has access to data, not modified while in transit, etc.

Transport service requirements: common apps

application	data loss	throughput	time sensitive?
file transfer/download	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5Kbps-1Mbps video:10Kbps-5Mbps	yes, 10's msec
streaming audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	Kbps+	yes, 10's msec
text messaging	no loss	elastic	yes and no

Services provided by TL

- mainly Two services

Reliable - data delivery
- TCP - Transmission Control protocol

unreliable BUT fast delivery.
- UPP - User Datagram Protocol.

TCP

- Connection-oriented, reliable data transfer.
- TCP will try to ensure - guaranteed delivery
  - delivery order
  - flow control (Not overwhelm receiver)
  - Congestion control (Not overload network)

A 3-way handshake before data transfer

- Stream of data. One-to-one comm^tn.

UPP

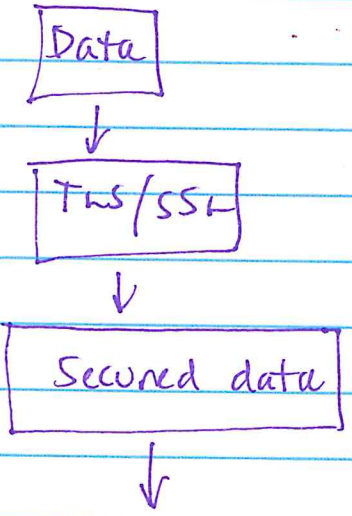
- No connection establishment
- No guarantee on delivery or delivery order
- Attempts to make the fastest delivery possible
- message based data. many-to-many comm^tn.

Internet transport protocols services

application	application layer protocol	transport protocol
file transfer/download	FTP [RFC 959]	TCP
e-mail	SMTP [RFC 5321]	TCP
Web documents	HTTP 1.1 [RFC 7320]	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary	TCP or UDP
streaming audio/video	HTTP [RFC 7320], DASH	TCP
interactive games	WOW, FPS (proprietary)	UDP or TCP

NOTE: No transport layer provides security services.

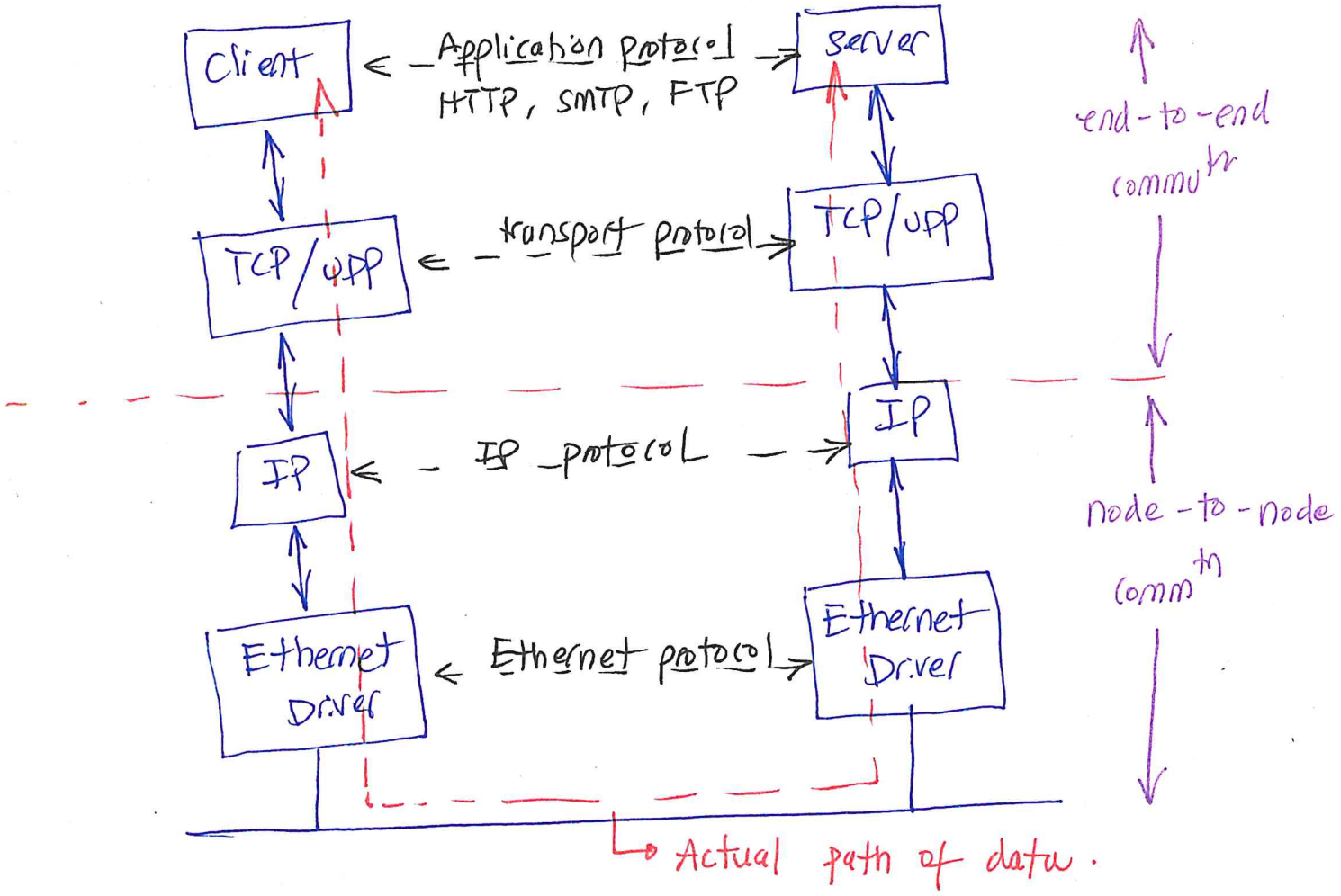
- Provided by an application-layer library called TLS
  - Transport Layer Security.
  - used to be called SSL.



- A third TL protocol called SCTP also exists
  - A Hybrid of TCP and UDP features.

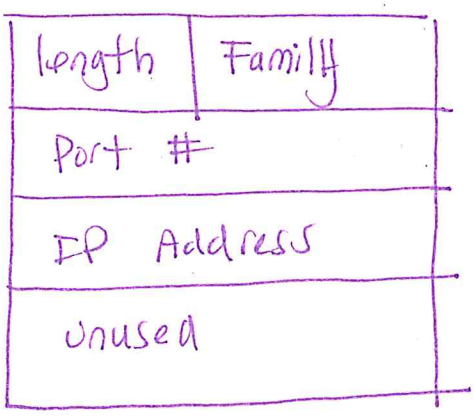
Stream Control Transmission Protocol.

Socket programming Tutorial

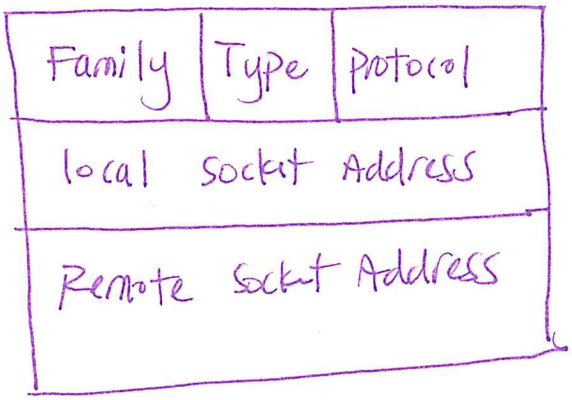


- A Socket exists only as long as a process holds a descriptor referring to it.

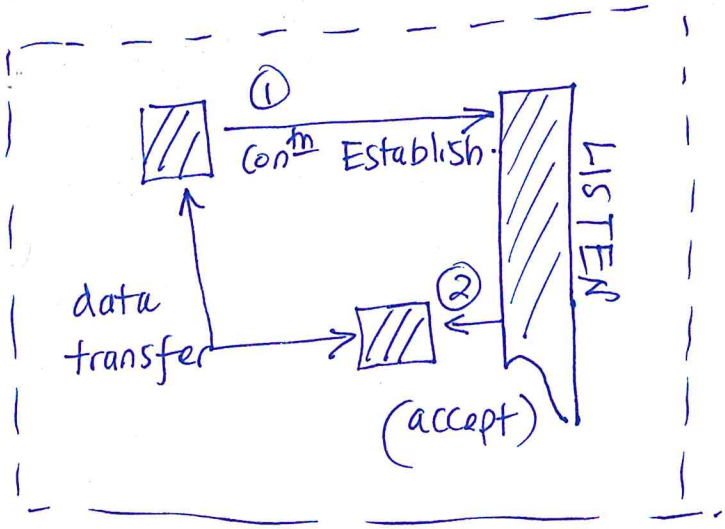
Socket Data structure



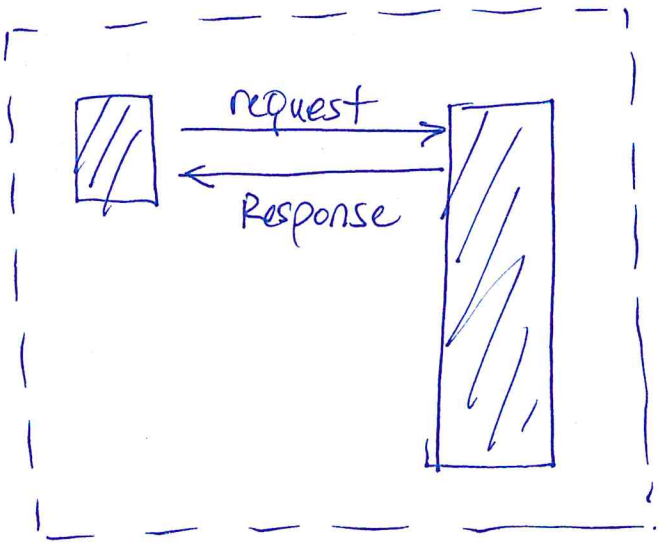
socket_addr

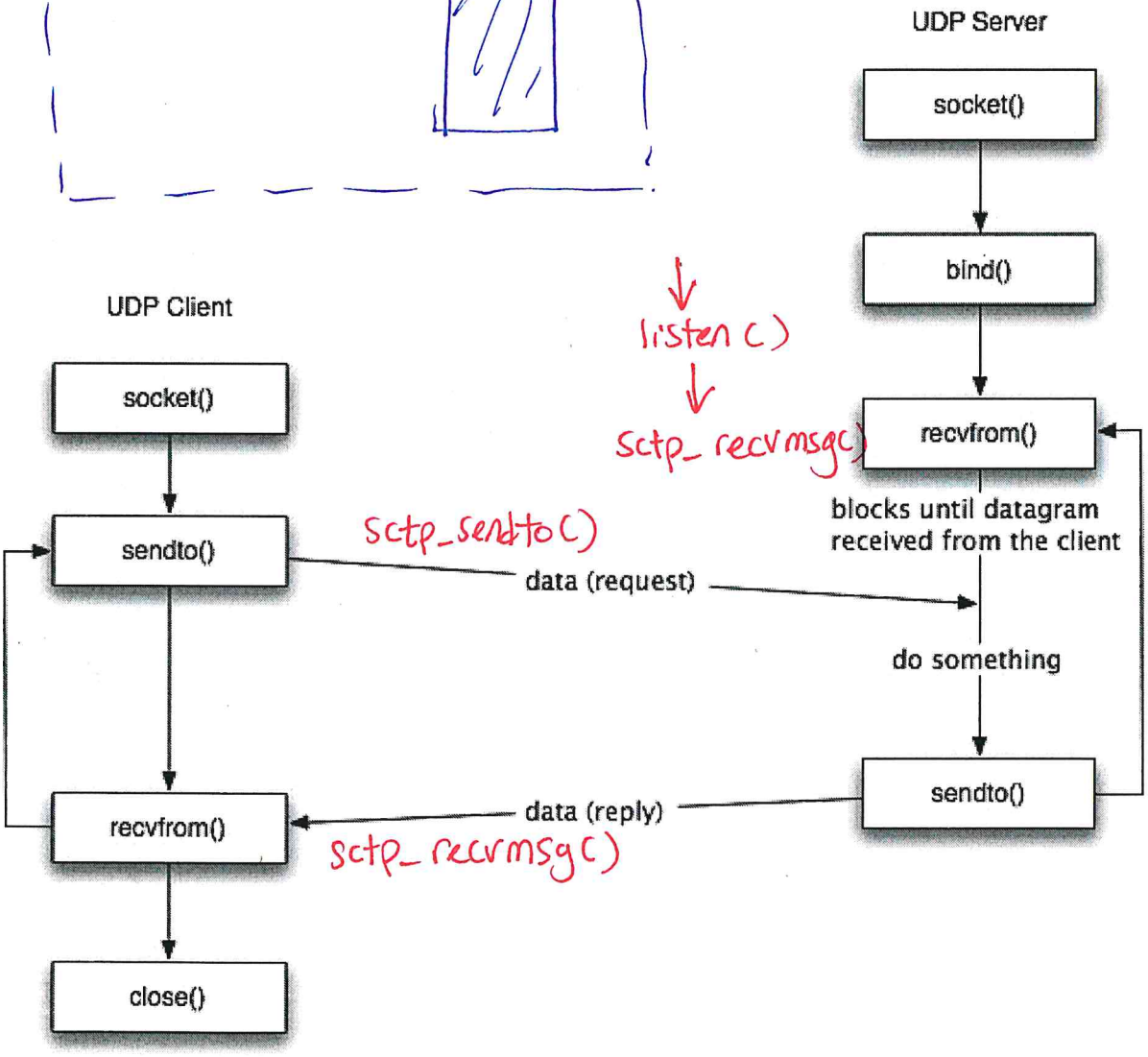


Socket



This model is similar to that of SCTP one-to-one Connⁿ.



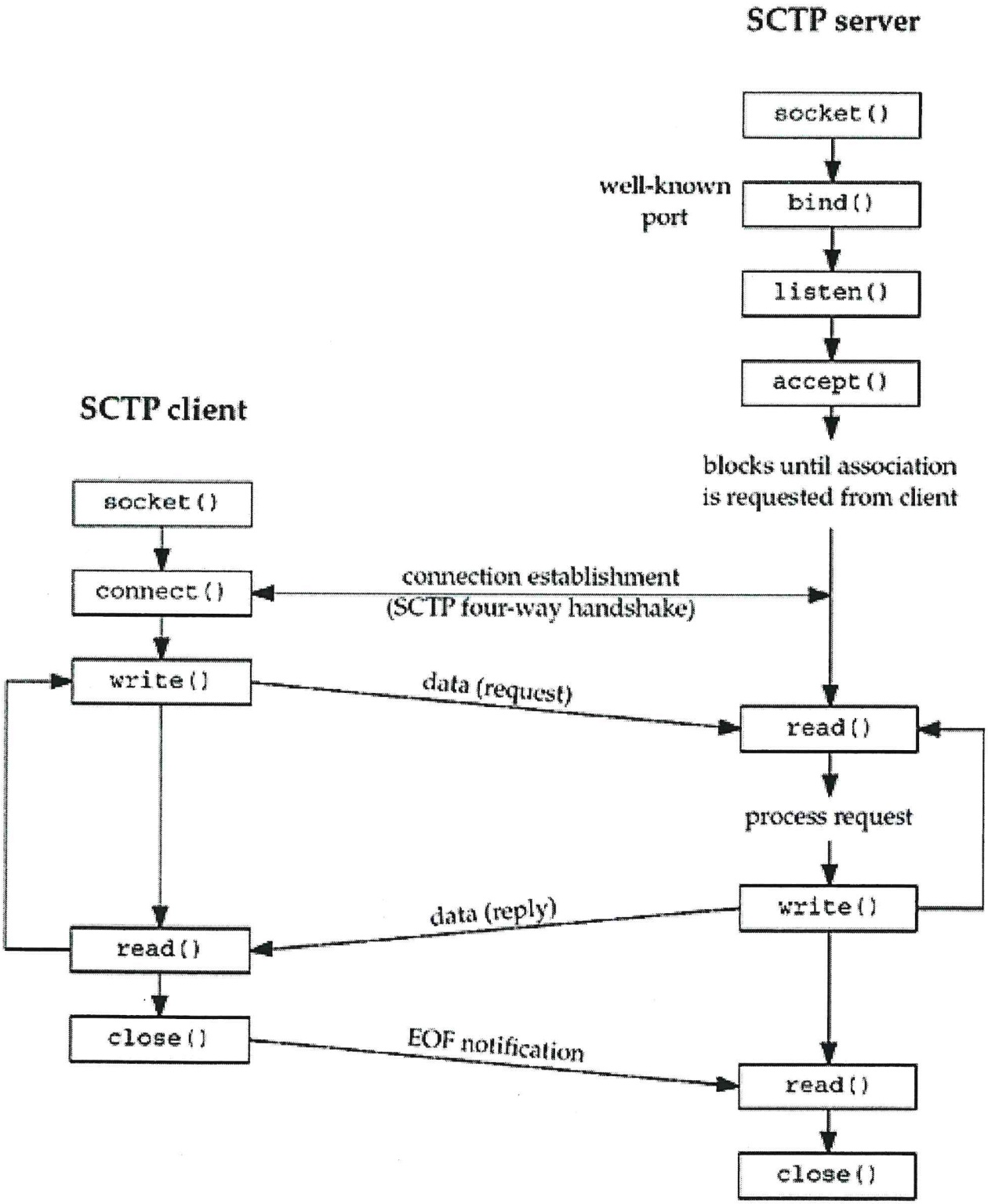


with highlighted modifications similar to sctp one-to-many connectn_

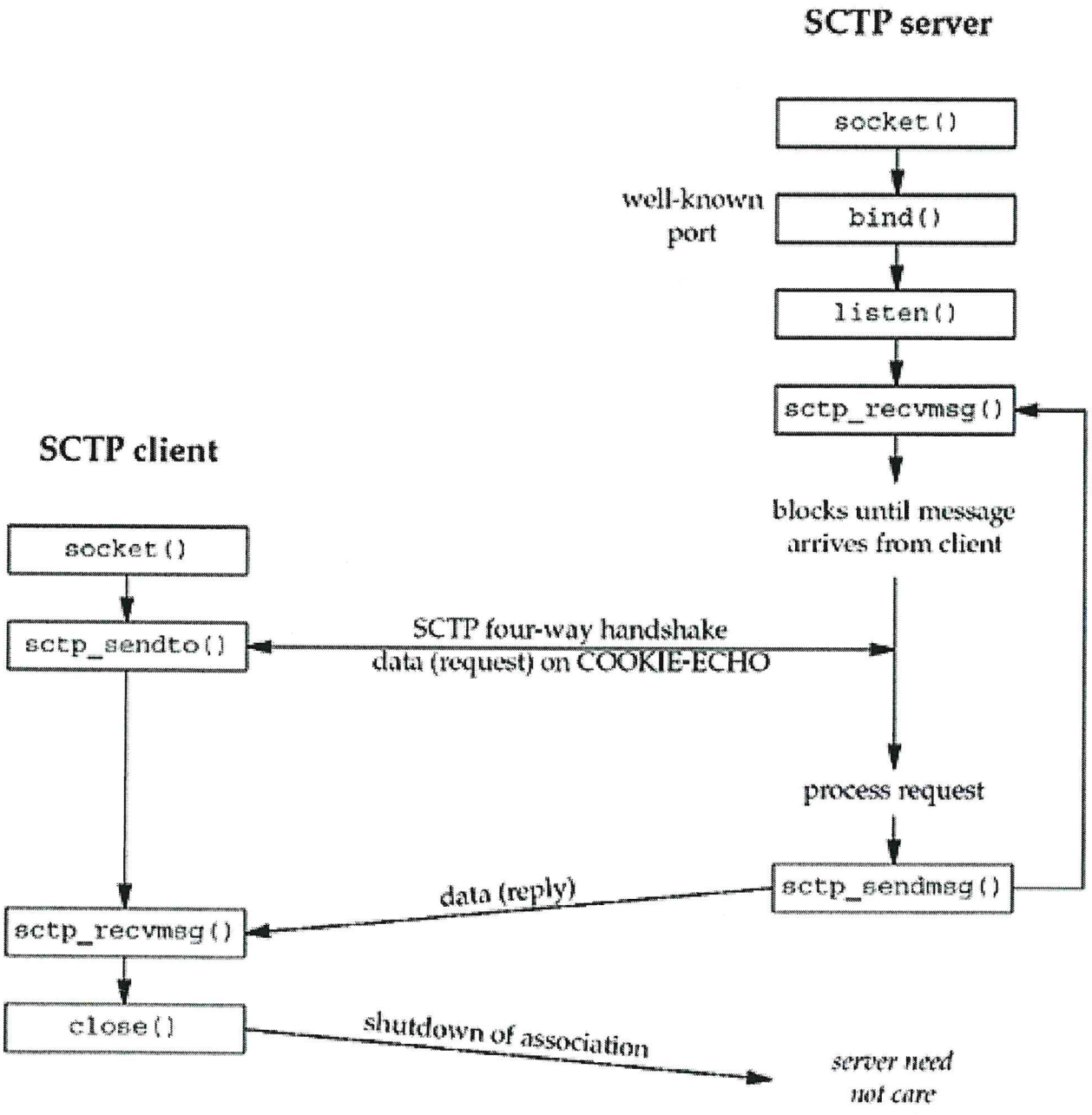
Socket API primitives

Socket	both	Create a socket and get the file descriptor.
bind	Server	Bind socket to IP + port addresses
listen	Server	Enter a "passive" listen state wait list incoming requests.
accept	Server	Accept incoming Conn^trs. initiate data transfer Conn^tn.
read / recv / recvfrom ↳ DGRAM (UDP)	both	receive messages.
write / send / sendto ↳ DGRAM (UDP)	both	Send messages
close		Terminate and release Port.
Connect	Client	Connect to remote (server) Socket (IP + port)

One-to-one - Association .



One-to-many Association.



Domain Name System (DNS)

- port 53 - Designed for UDP. (Responds to TCP too.)
- RFC #1034 / #1035
- An auxillary protocol
  - Helper protocol → "speed" over reliability → UDP.
- Primary Function: Convert Hostname → IP Address

Hostname: Human Readable Format

IP Address: Machine Readable Format

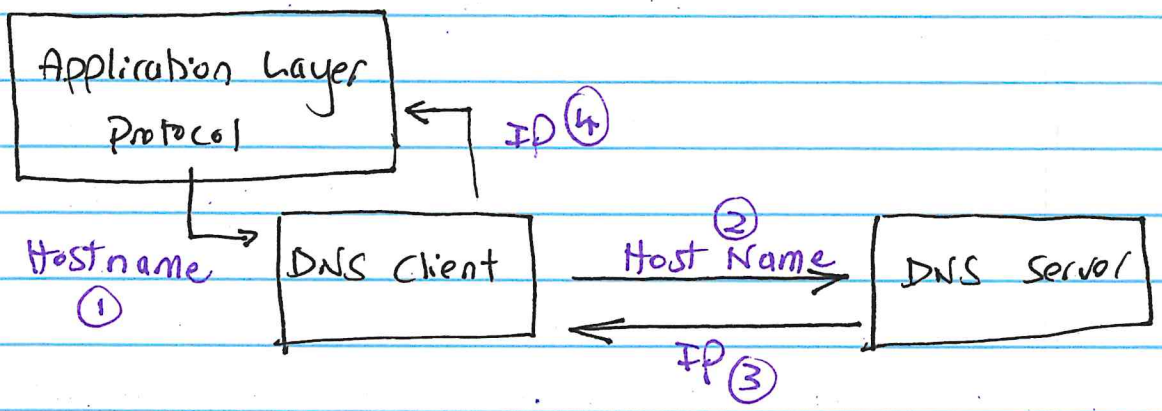
A Directory Service hierarchy
- provides hostname → IP Address mappings.

Hostnames: Variable length, different string formats, human readable.

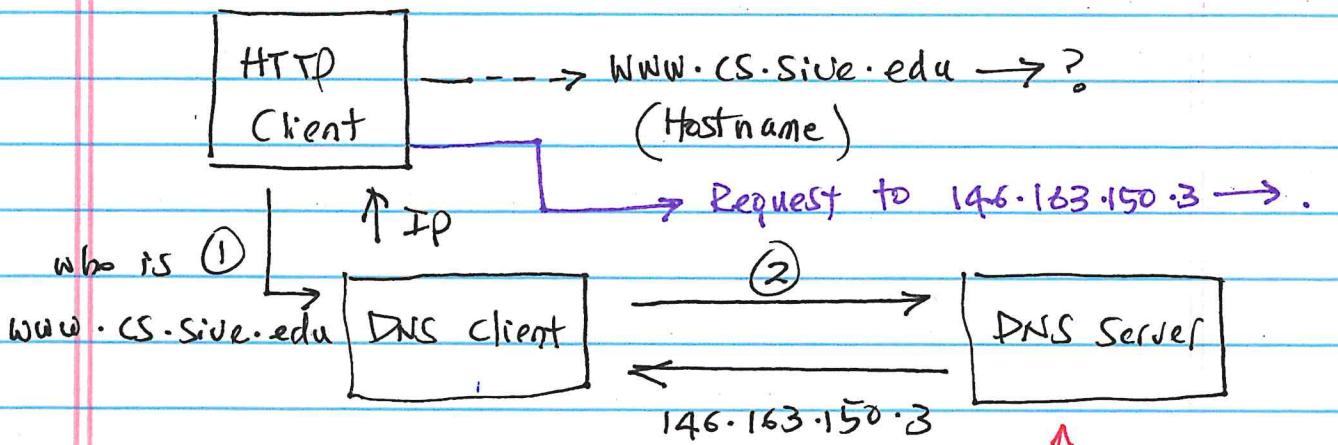
IP Address: Fixed size and format, machine readable.

e.g. IPV4 127.0.0.1 (32-bits) — dotted decimal Notation.
IPV6 (128-bits)

Network CORE prefer IP addresses for faster indexing.



Example:



- Domain name space is hierarchical
  - → within a name space, each hostname is unique
  - → within a domain, each IP address is also unique.

- A fully qualified domain name (FQDN) indicates the hierarchy.

e.g. eb3053.cs.sive.edu.

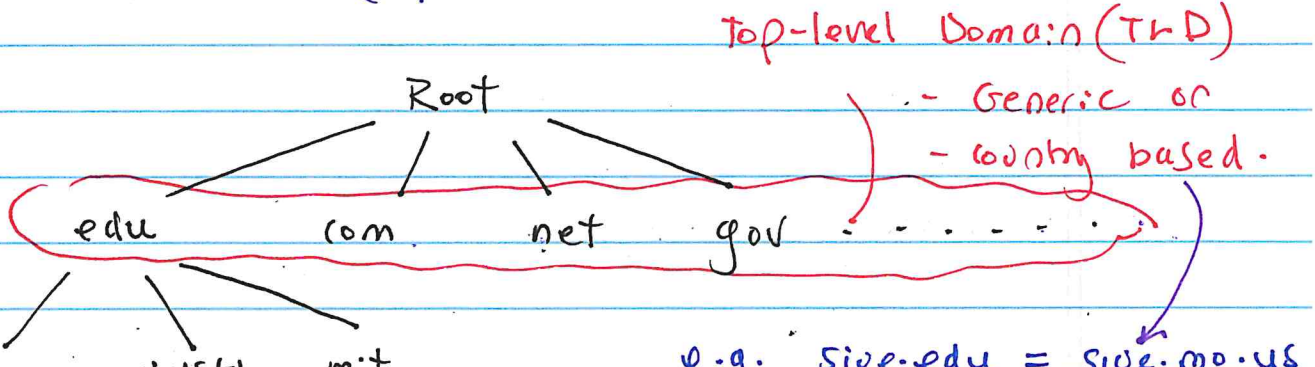
cs domain

edu domain.

Host name (label)

sive domain

- Hostname by itself is a partially qualified domain name (PQDN)



- PQDN can be resolved within the domain

- e.g. "eb3053." can be resolved within cs

"eb3053.cs." can be resolved within sive

"eb3053.cs.sive." can be resolved within edu etc.

Q: why a hierarchy?

- A single server can't keep all mappings
- Mappings can change, IP can be re-assigned
- Not efficient or reliable to store all DNS mappings in a single server.

> - Thus DNS uses a distributed, replicated DNS server hierarchy

- A DNS client is called a "resolver"
  - Resolvers access the nearest DNS server for mappings
    - A client-server interaction.

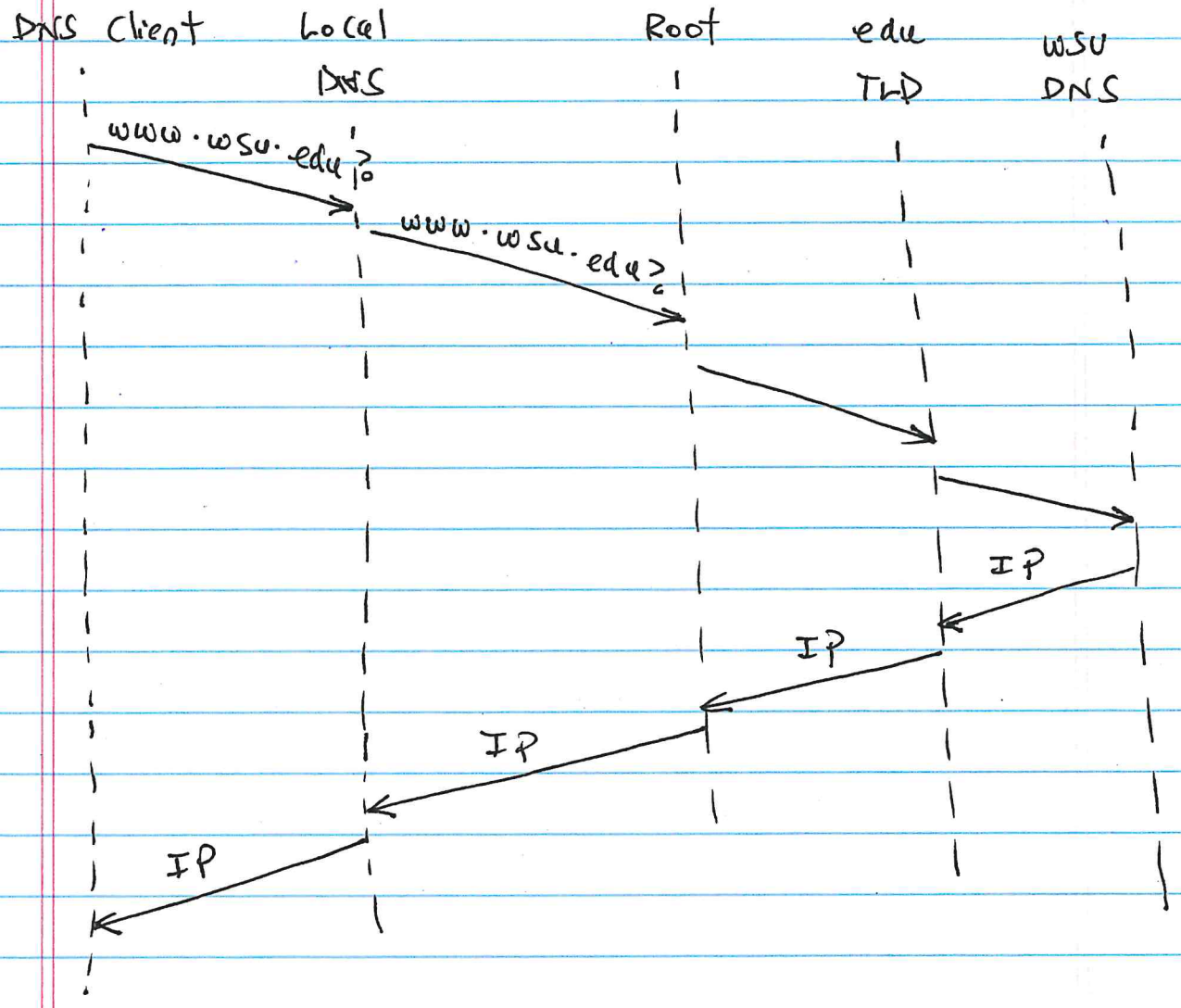
- Two types of resolutions
  (i) Recursive

  (ii) Iterative.

- On Linux /etc/resolv.conf lists nameserver info.

Recursive DNS resolution.

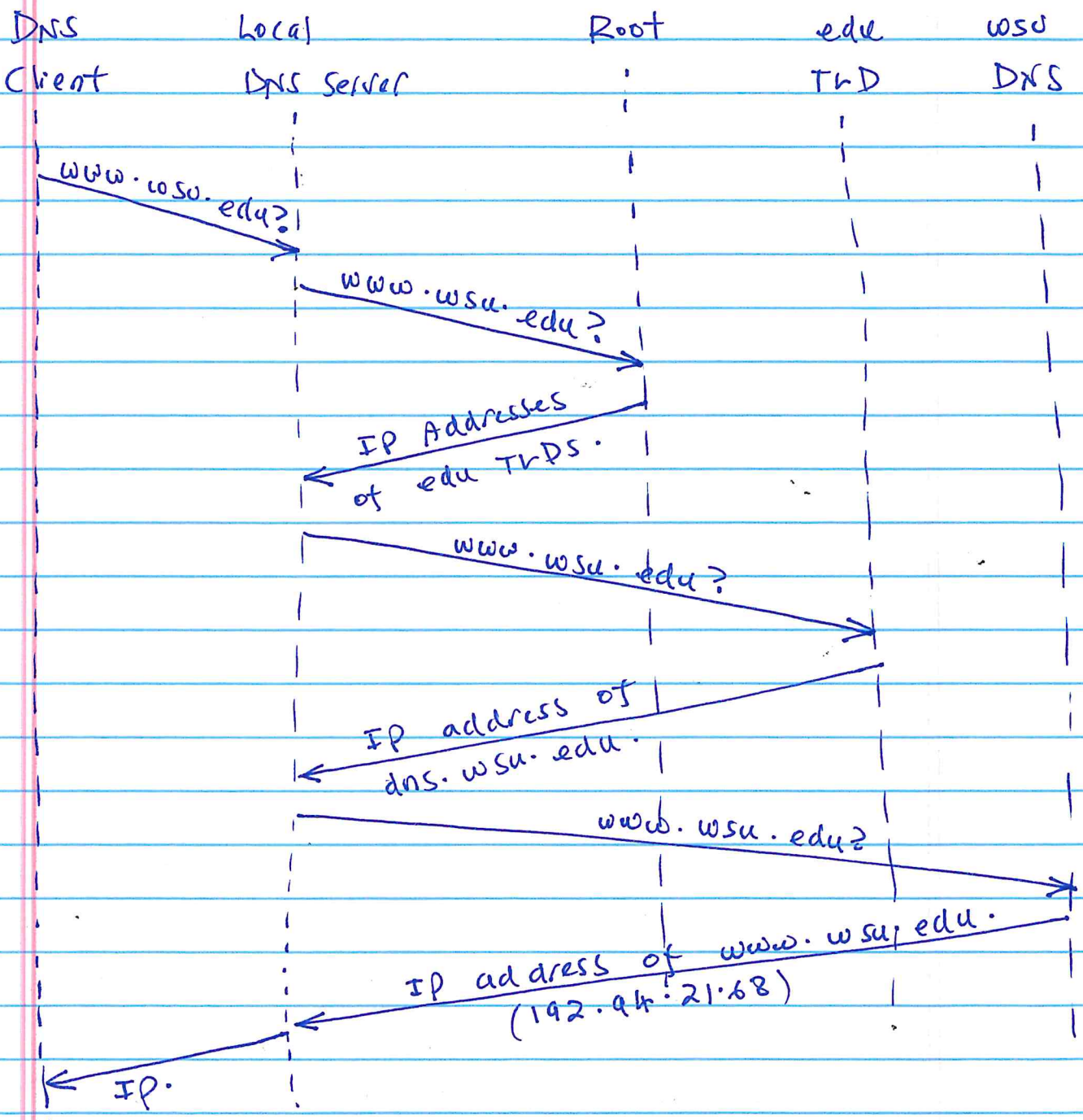
e.g. A machine in Sive domain looking for "www.wsu.edu" DNS.



- Root does not know "www.wsu." but knows about edu. → ASK from edu.
- edu TLD server knows ~~wus~~ "wsu" but not "www". ASKS ~~wus~~ wsu DNS server.

Iterative

~~Recursive~~ DNS Resolution



- DNS is a binary protocol (Not a plain text protocol like SMTP, HTTP).

- the 'host' command is a DNS lookup ~~util~~ utility on Linux.
  - other tools: nslookup, dig.
- DNS is cached locally. A subsequent request can be fulfilled from the local cache.
  - On linux nscd service can be used to cache DNS lookups.
    - stored in /var/cache/nscd/

- On windows "ipconfig /flushdns" can be used to flush cache.

Application Layer Protocols

HTTP : Hypertext Transfer Protocol [Port 80] [+SSL Port 443]

- Runs over TCP.
  - two conns. Control and data.

- webpage : An object stored in a web server with possible links to other objects.

  → each object can be uniquely identified (to distinguish from other objects)
    - there are many web pages, hence many objects.

  - each object is a file addressible by a single URL.

URL — Uniform Resource locator.

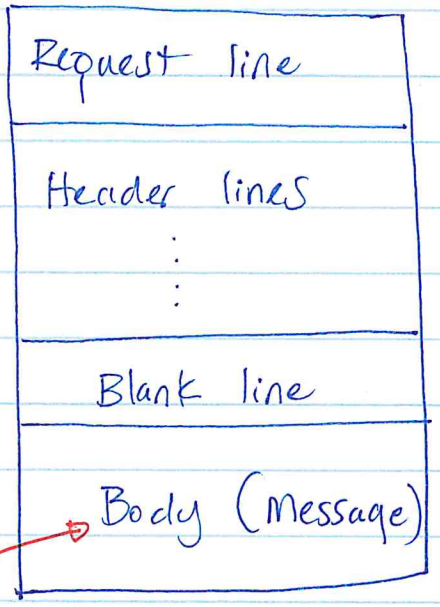
URL ≡ protocol + Host + port + path.

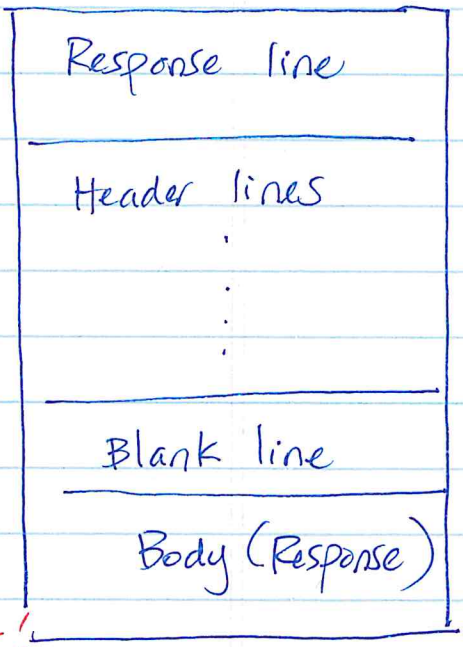
↑ typically ignored since protocol defines the port.

e.g. http://www.cs.siue.edu/~tyumage/index.html

- Follows a request / response format for exchange.

HTTP message Format





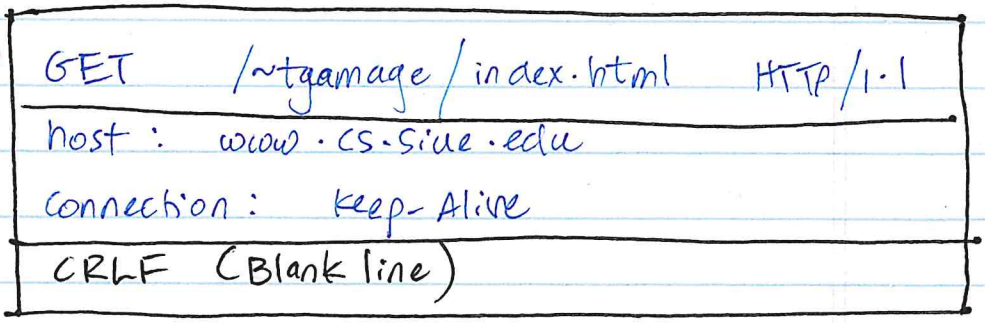
empty in GET messages; populated in POST!

Request

Response

Request line

e.g.

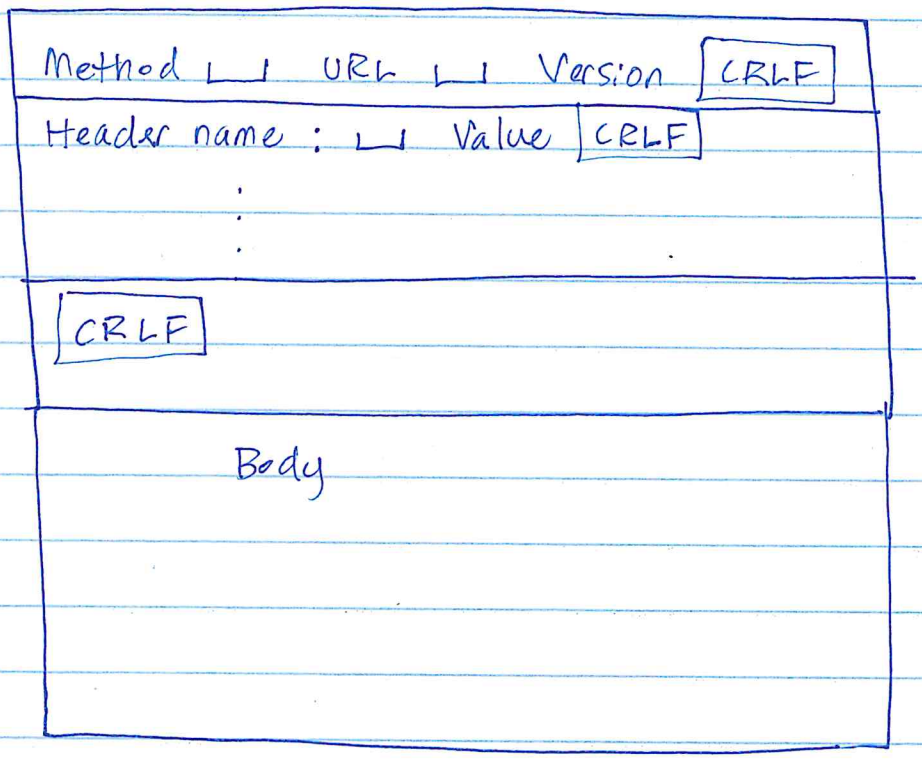


Headers

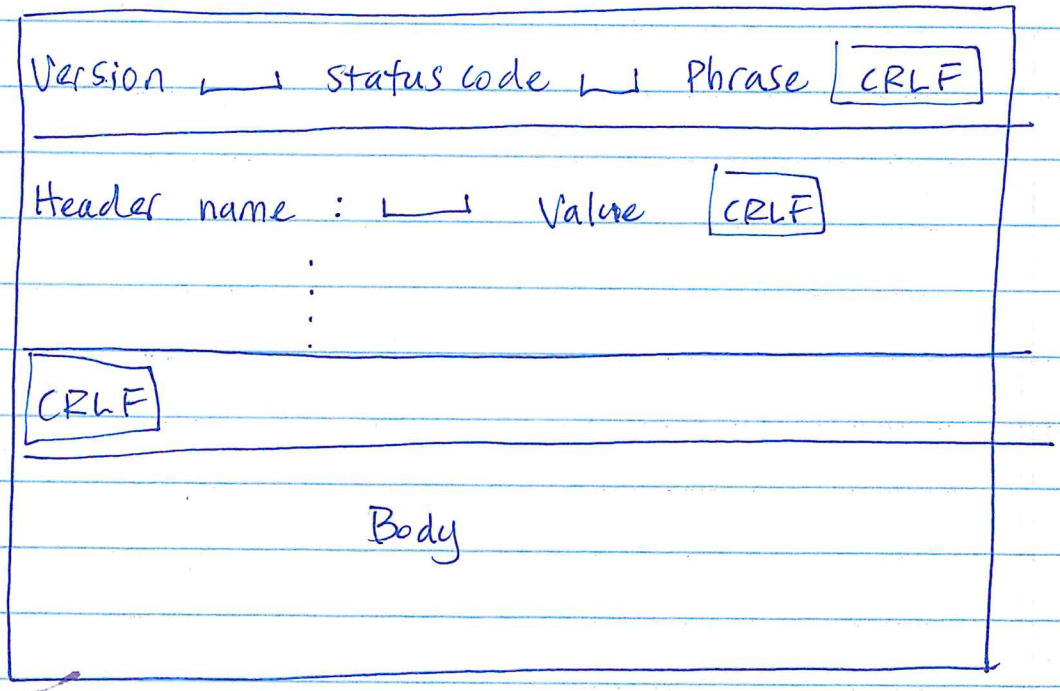
Blank line

No body.

Request



Response



demo with following Commands.

telnet www.cs.siue.edu 80

(i) GET /~tgamage HTTP/1.1
~~host : www.cs.siue.edu.~~

→ Bad request 400.

(ii) GET /~tgamage HTTP/1.1
host : www.cs.siue.edu.

→ Moved permently 301

(iii) GET /~tgamage/index.html HTTP/1.1
host : www.cs.siue.edu

→ OK 200.

Q: where can you learn about methods and, Headers, response codes ?

→ Read the ~~artificio~~ [RFC 2616.]

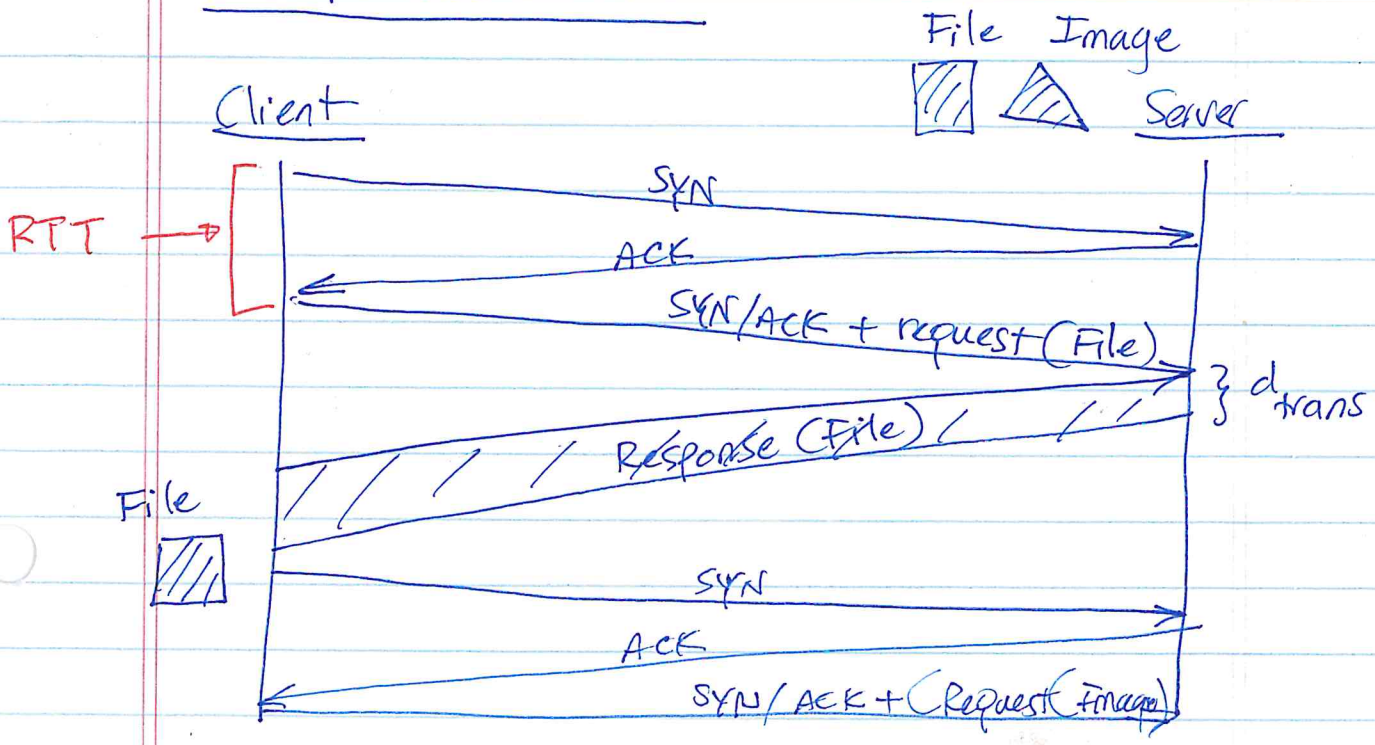
- Every object is uniquely addressible.

e.g. A web page with 5 JPEG Images
↳ A total of 6 objects.
5 images + HTML File.

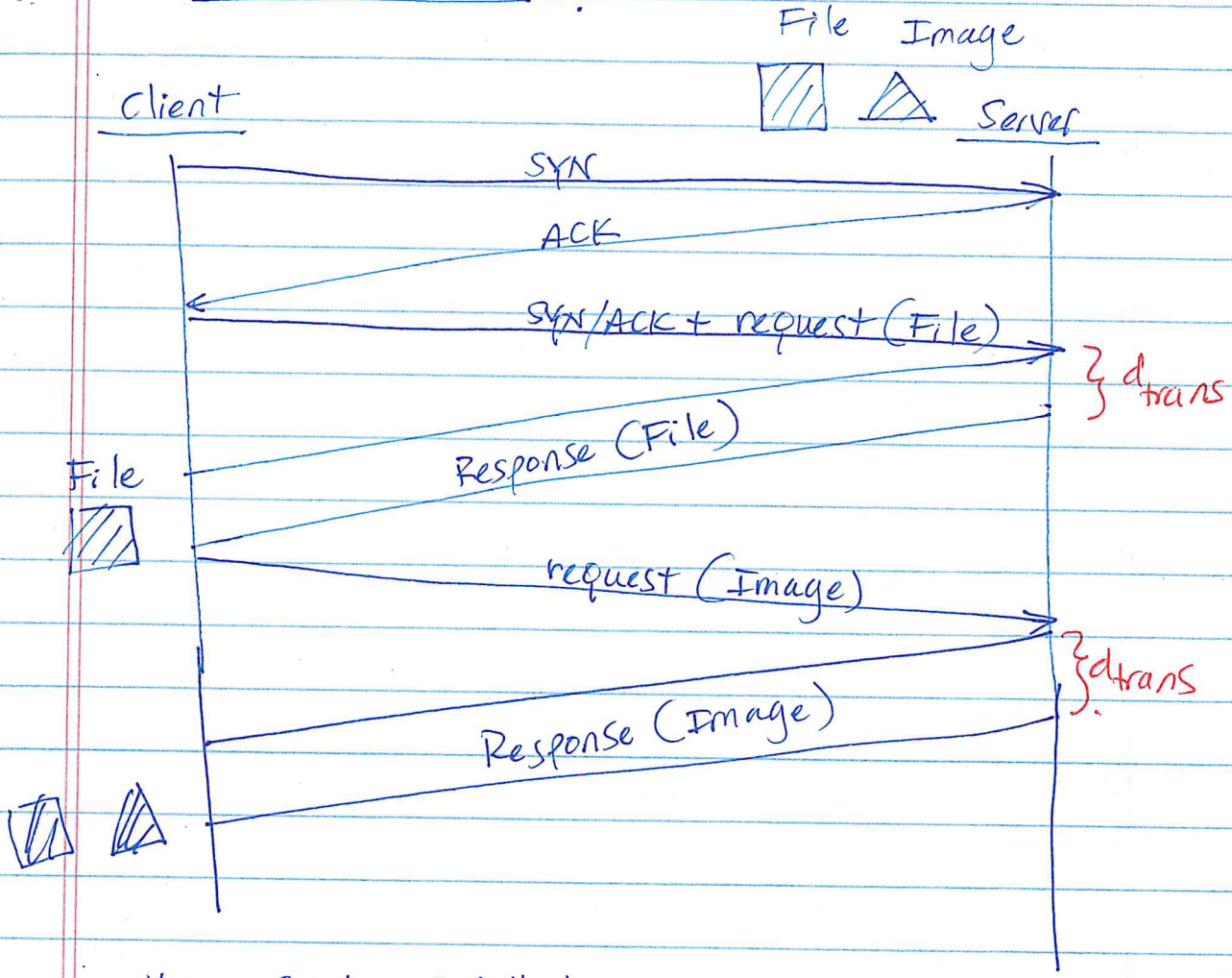
Retrieval.

1. Assume all objects reside in the same server.
2. Two options:
   - ↳ Retrieve through a single TCP conn.
     Persistent conn: default in HTTP 1.1
   - ↳ Retrive through seperate TCP conns.
     Non-persistent conn: prior to HTTP 1.1.
     ← (Resource Hog).

Non-persistent Conns



Persistent con^tn.



"Connection" ~~token~~. Header token.

Connection: close — Non-persistent

Connection: keep-alive — persistent

- Modern browsers typically can make several concorrent requests.

Chrome : 6

Firefox : 6

IE 11 : 13

Source: browserscope.org.

HTTP is Stateless

→ The server does not remember (or keep a memory of) the client or client request

└ Client requests → Server responses → done.

→ Ask for the same object in 2secs; server sends again.

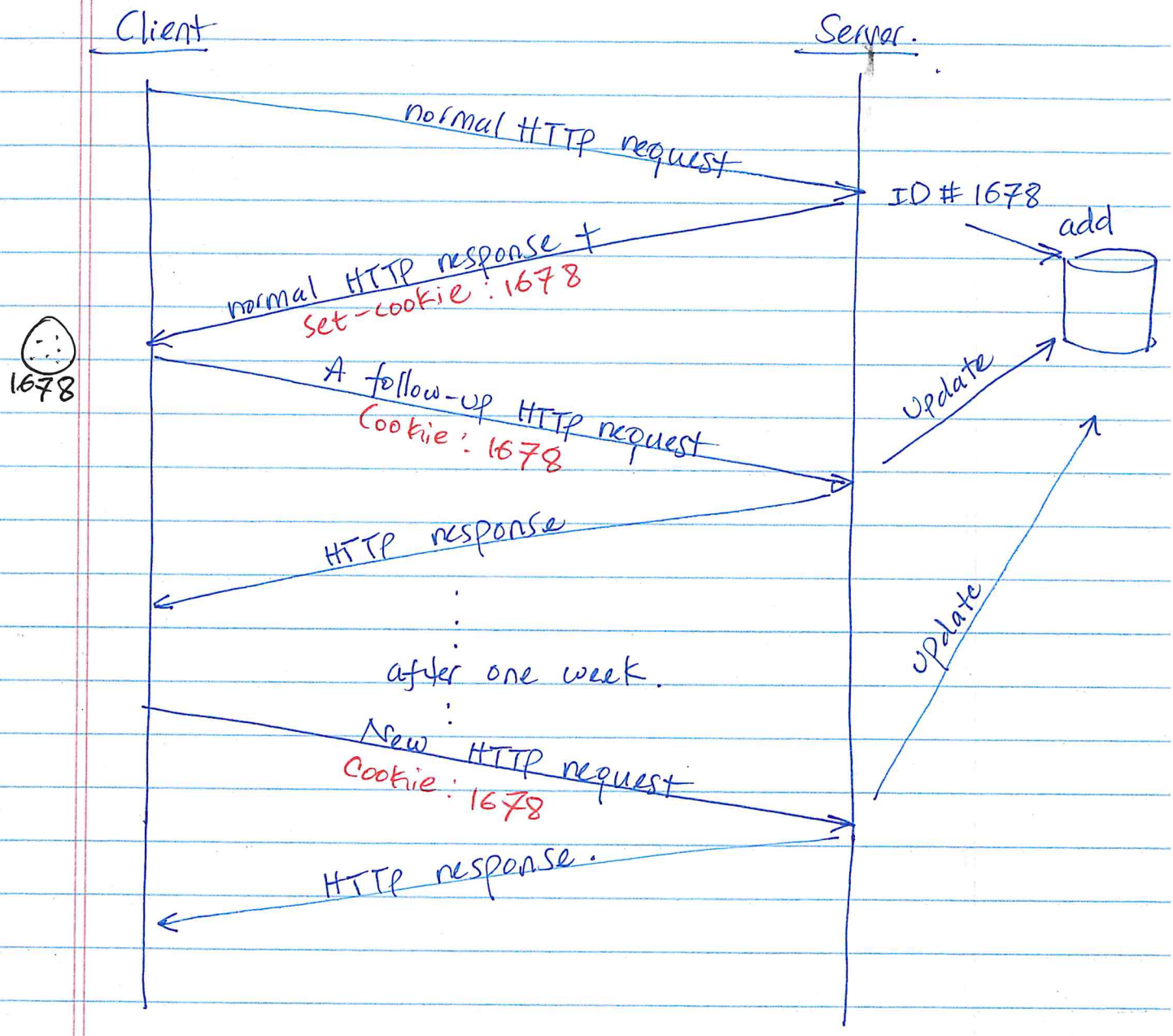
i. Advantages: Simplifies server design
Better performance + Allows Simultaneous Conn^ns.

ii. Disadvantages: Registered Clients (For repeat Access)
Online shopping (recent interests)
Block users / IP addresses.

Solution: Cookies. [RFC #6265]

- Allow web sites to keep track of users.

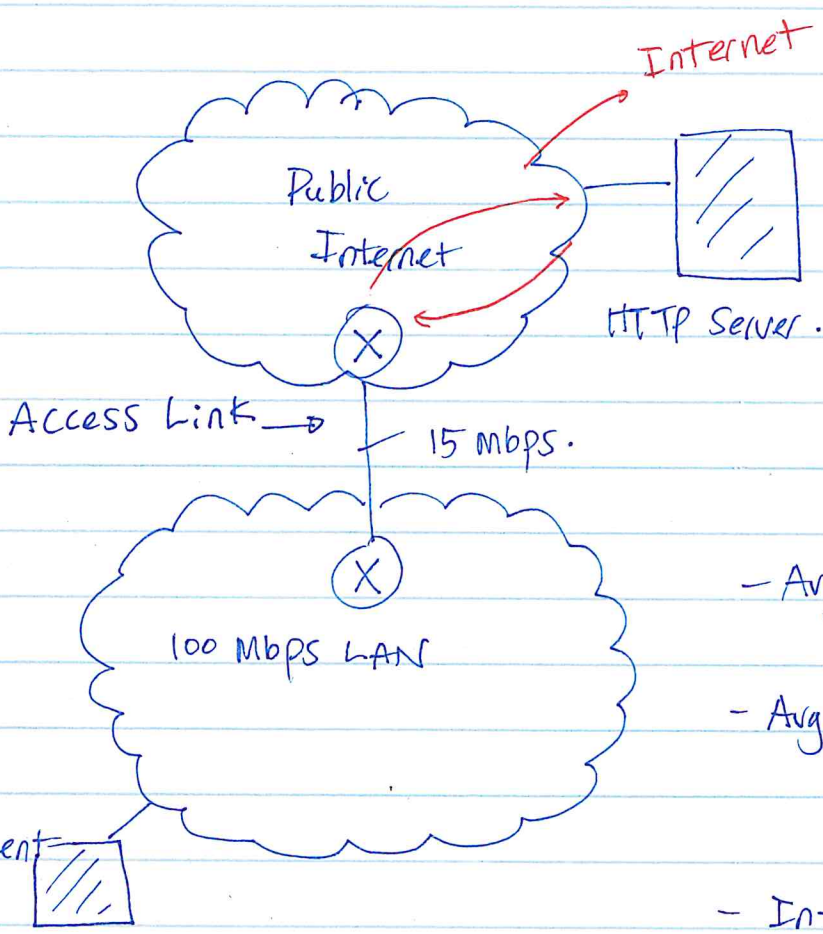
- 4 Components
  - A cookie file managed by Client browser
  - A Back-end DB with user info @ server
  - @: 'Set-cookie' header on response
  - 'Cookie' header on subsequent requests.



- You can ask your browser not to set cookies.
  → Chrome: "Do not track"

Web Caching

Consider the following setup.



- Avg. object size = 1 Mb
- Avg. request = 15/sec.
- Internet Delay = 2 secs.

Calculations:

(i) LAN traffic intensity

$$15 \frac{\text{req}}{\text{sec}} \times \frac{1 \times 10^6 \text{ bits}}{\text{req}} \times \frac{1}{100 \times 10^6 \frac{\text{bits}}{\text{sec}}} = 0.15$$

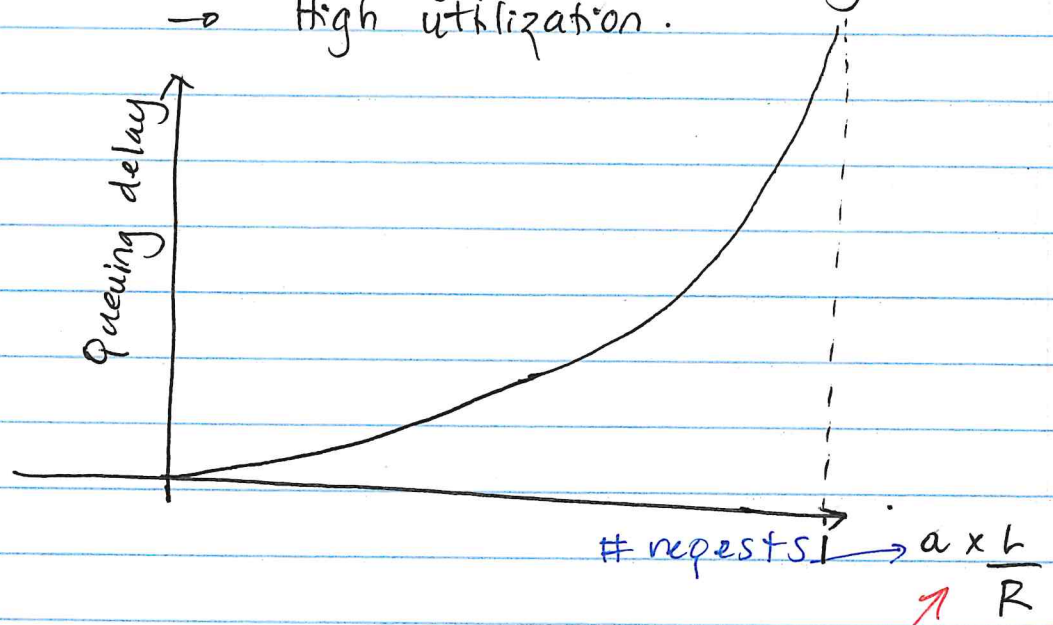
- low traffic intensity.

(ii) Traffic intensity of Access Link.

$$\frac{15 \text{ req}}{\text{sec}} \times \frac{1 \text{ Mbits}}{\text{req.}} \times \frac{1}{\frac{15 \text{ Mbits}}{\text{sec}}} = 1$$

Bad.

⇒ Access Link already @ full capacity.
→ High utilization.



traffic intensity.

- delay grows w/o bounds. (order of minutes).

(iii) Total response time

LAN delay + access delay + Internet delay.

negligible (low traffic intensity)

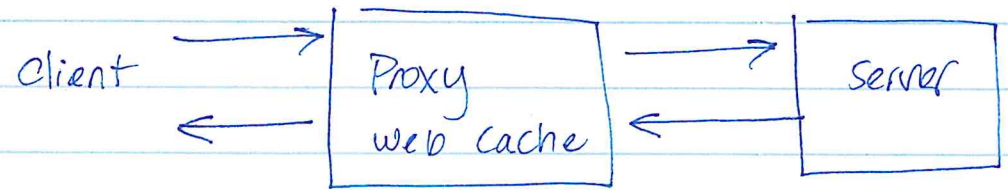
proportional to queuing delay.

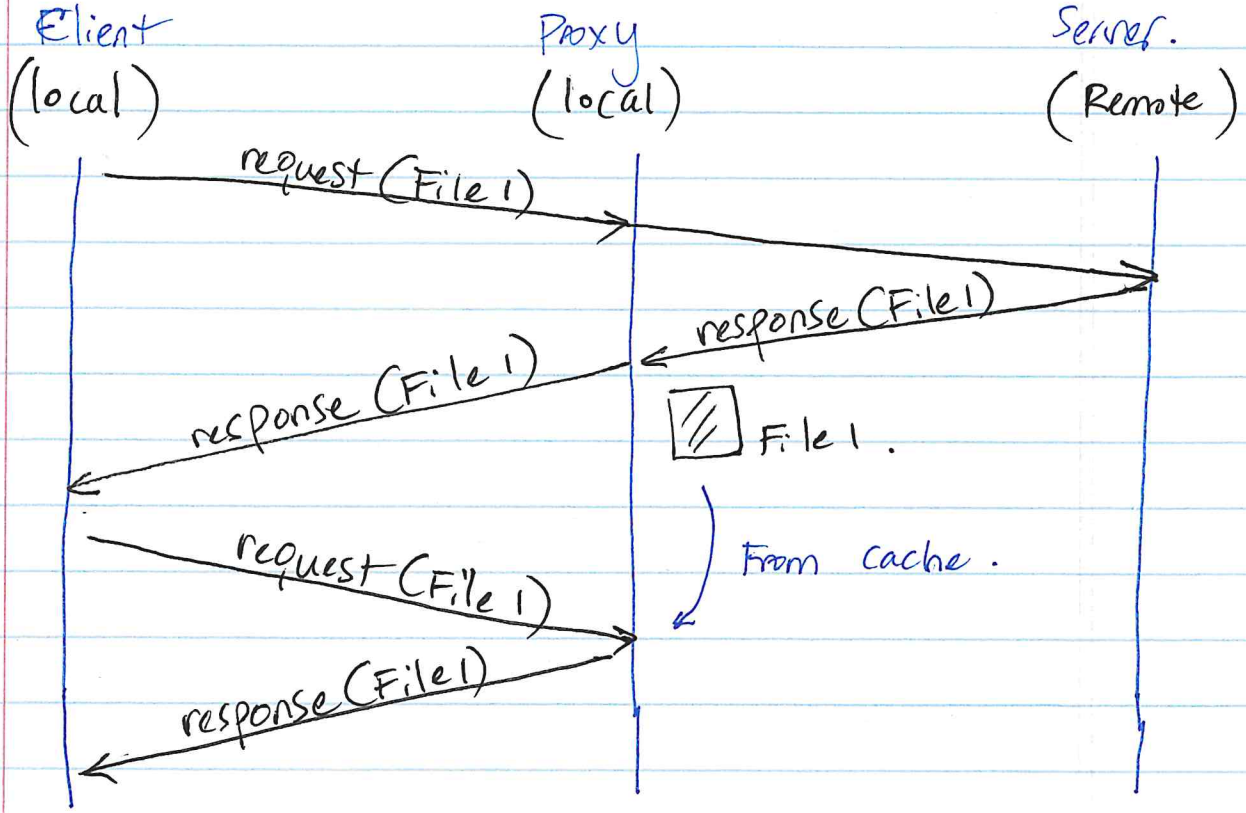
2 secs.

Solutions : Ask Students.

(i) Infrastructure Investment Cost
- → Bump Access link 15 Mbps → 100 mbps.
- → Costly to the institutions.

(ii) Use a Web Cache
- → A local proxy storage between client and server.
  - → A.K.A. forward proxy.





Recalculations:

- Typical cache hit rates : 0.2 ~~0.1~~ – 0.7
  - Assume 0.4 here.
  - 40% request served locally
  - 60% (cache miss) goes to public Internet

- Traffic Intensity on Access link

  1 —drop→ 0.6.

  - Typically a traffic intensity of $\gg 0.8$ is considered small delay
    - Assume 10 msecs here.

Response time

LAN delay + Access delay + Internet delay.
(Assume 10 msec)

$$0.4\left(10 \times 10^{-3}\right) + 0.6\left(2.01\right)$$
$$= 1.2 \text{ secs.}$$

- Forward proxy is for clients
- Reverse proxy is for servers. (A proxy for servers for security, authentication, reliability, performance).

Any down sides?

→ what if the original object (web page) changes?

→ cache might not be up-to-date - stale data.

So!: Conditional GET request.

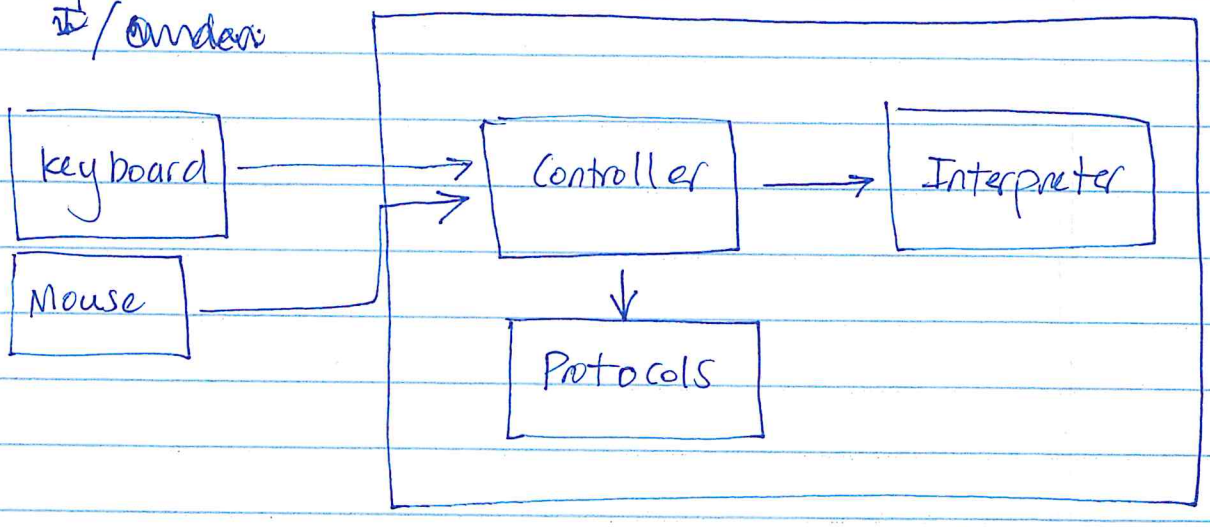
Add a Header If-modified-since: <time>

A Side note on web browsers.

- Acts as a client for HTTP Comm^{tn}.
- Not the only protocol it can work with.
- Three components: Controller + protocols + Interpreter.

Browser

e.g. ftp://ftp.adobe.com



- Controller takes input from Input device
- Supports many client protocols
  e.g. HTTP, FTP, SSH, SMTP.

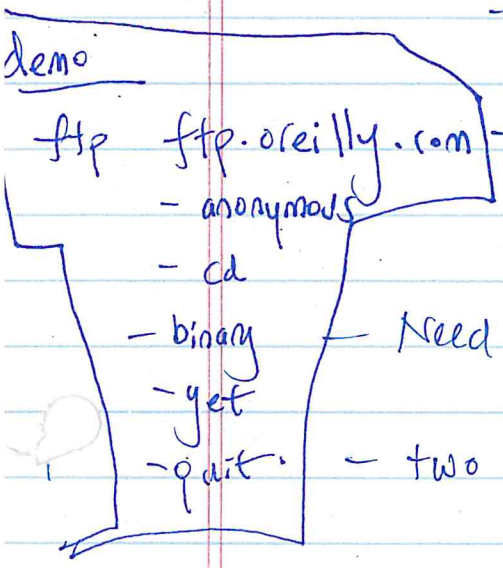
- Interpreter : Depending On the Content received and protocol displays results

  e.g. HTML, JAVAscript, java.

File Transfer Protocol (FTP) [RFC 959]

[Port 21]

- Simple and straightforward process (on the surface)
- Complexities
  - possibility of different file name conversions
  - Different ways to represent data
  - Different directory structures.
- Need common agreements; hence, FTP.



- two Connections — out-of-band connection.
  - Control Connection over port 21
  - Data Connection over port 20

↑ server side.

- Control Connection remains open for the entire session
- Data Connection opens and closes per file transfer.

demo: ftp ftp.adobe.com.

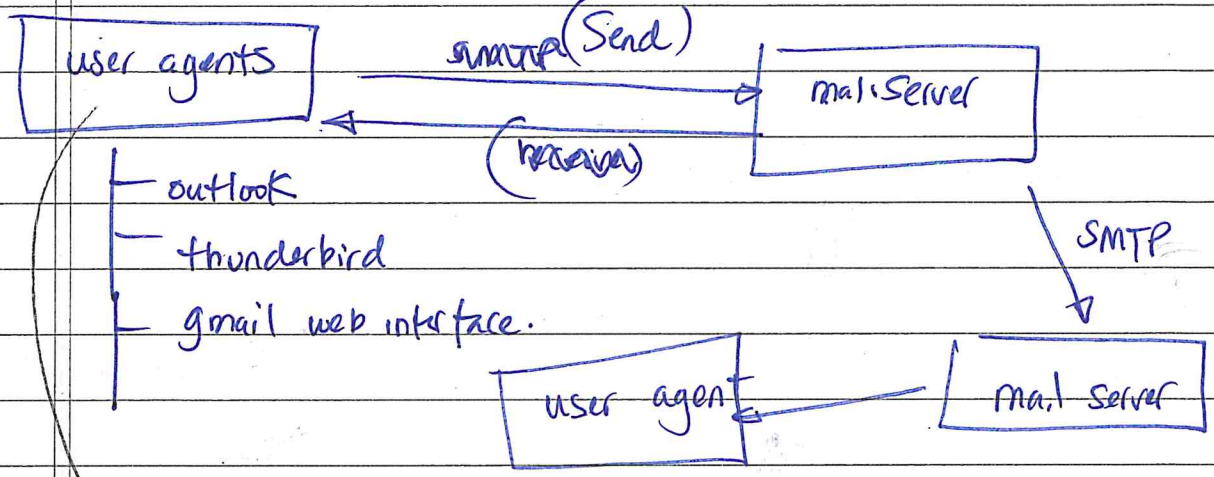
telnet ftp.adobe.com 21
(can't list or retrieve).

use the browser to show

09/17/14

Electronic Mail

- Asynchronous Communication Medium
  - what is Asynchronous mean?
    - Two parties need not be online at the same time.
- Runs over TCP.



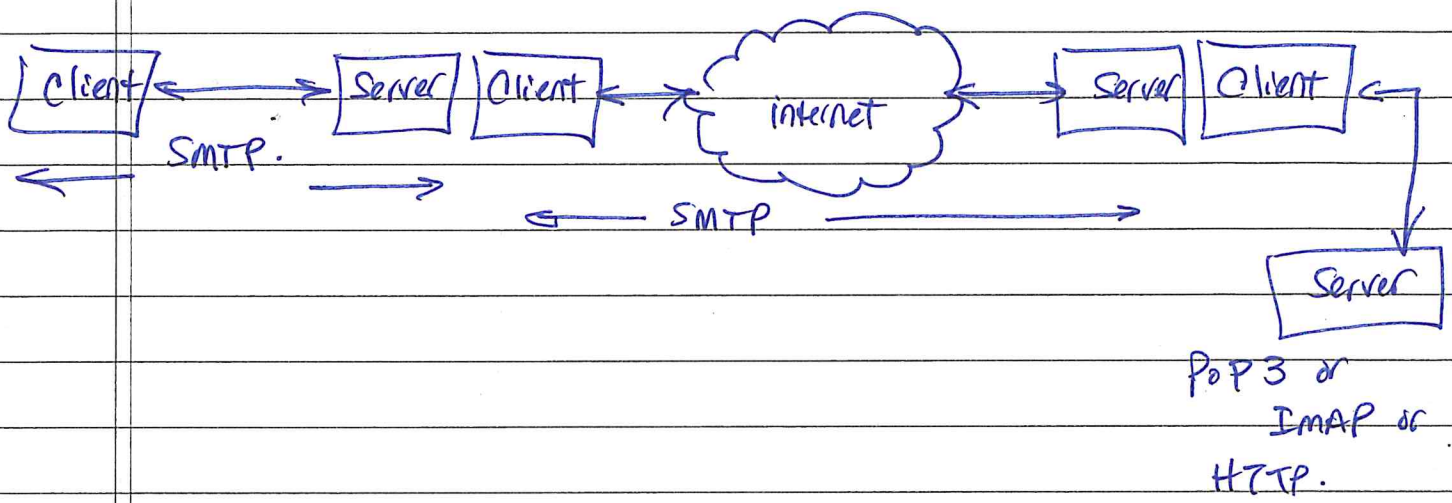
- allows users to read, reply to, forward
  delete, organize, etc...
  Compose.

each mail server holds a mailbox for each user it serves.

- gmail
- hotmail
- yahoo
- Sjue.

7.1

email has three pairs of client-server communication.



MIME- Multipurpose ~~Intrac~~ Internet Mail Extension.

SMTP uses (forces to use) 7-bit ASCII.

— MIME is a supplementary protocol that allows non-ASCII data to be send through email.

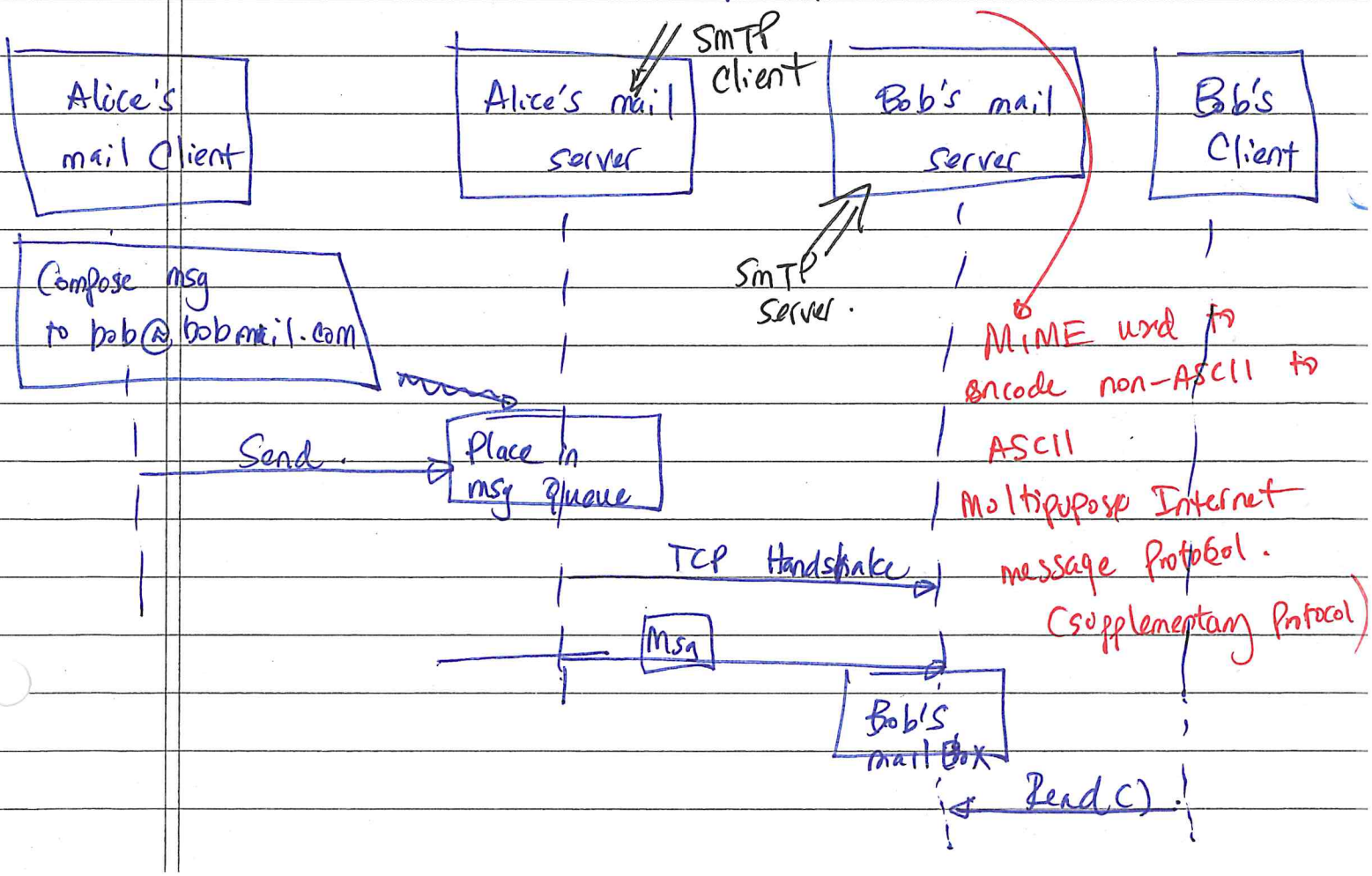
SMTP : Simple Mail Tranfer Protocol.

[Port 25] [RFC #5321]
[TSSL port 465]

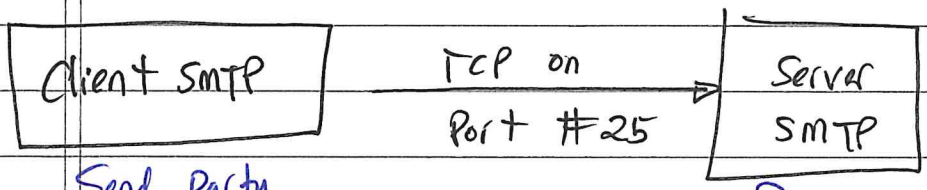
- Transfers messages from Sender's mail server to receivers mail server.
  - message exchanges between mail servers.
  ~~Not between~~
- Older than HTTP.

- Restrictions (legacy Technology)

Restricts the body (not just the header) to be simple 7-bi ASCII
- images, audio, video files all needs to be ASCII encoded.



→ No intermediate mail servers. Directly communicates with the corresponding receiving mail server.



Send party
CLIENT

Receive party.
SERVER.

→ If server is down retry again

Compare Compare with HTTP.

→ Both protocols transfer files from one host to another.
HTTP: objects SMTP: files (email, messages)

→ Both persistent HTTP and SMTP use persistent connⁿ.

→ HTTP pulls. SMTP pushes.

→ SMTP require 7-bit ASCII encoding.

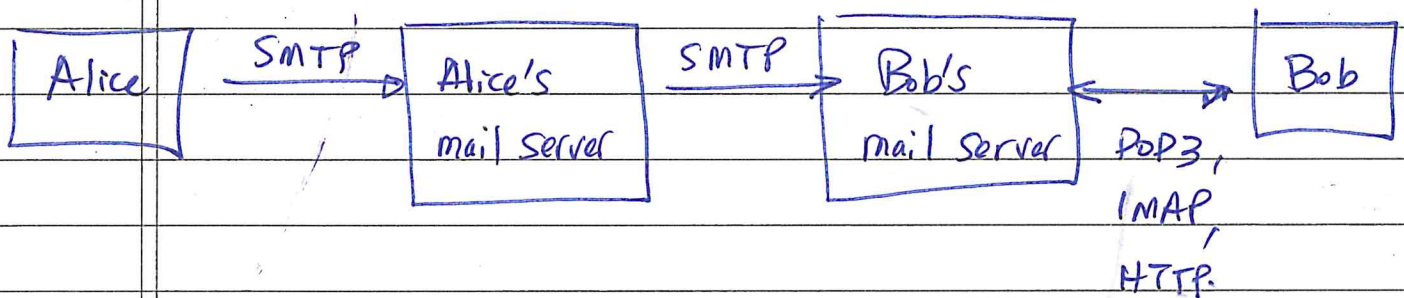
→ HTTP: each object in seperate HTTP response
SMTP: everything in one message.

We've talked about mail delivery. what about mail access?

Mail Access Protocol

→ Naturally it's easy to think bob's email server to run on Bob's PC
→ Can do ~~must~~ BUT MUST be online all the time

So: Run a mail client and access the server on-demand.



POP3 → Post Office Protocol - ver. 3 [Older Standard] [RFC #1939]

- A TCP Connection over port 110. [+SSL: port 995]

Authorization : username and password sent (in the clear) to authenticate the user.
↓
Transaction : Retrieve messages; can also mark delete, ~~remove~~ undelete, labeling, etc.
↓
Update : ~~Quit~~ after Quit (update according to the user actions).

~~IMAP - Internet Mail Access Protocol.~~

POP3 ~~creates a folder~~ downloads message to local machine (if using a POP3 desktop client)

→ Danger → If download-and-delete, will lose access from other clients.

Can't create remote folders and assign messages to folders.

[port 143] [+SSL port 993]

IMAP - Internet Mail Access Protocol. [RFC 3501]

→ Much more functional than POP3. also more complex.

- POP3 does not allow remote folder maintenance: Arrange folders, assign msgs to folders, etc.

→ Solved in IMAP.

→ Also allows (has support) components of a message

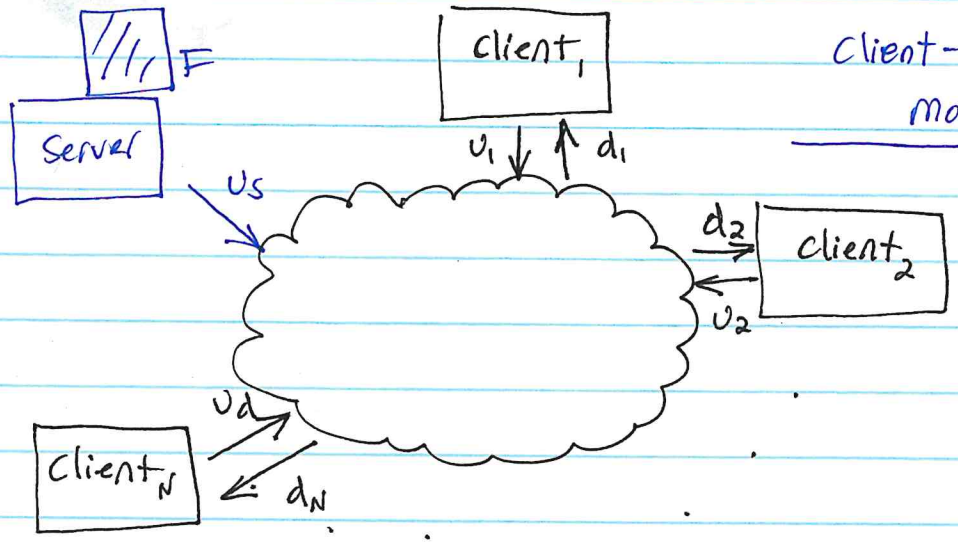
e.g. get only headers.

P2P Systems

- AL - level resource organization for flexible sharing
- Each "peer" acts both as a client and a server.
- provides a large combined storage, cpu power, bandw. dth, etc.
- low cost on scalability : scales with # of peers
- Self managing (correct oper^{tn} is not dependent on a central server), search not bottlenecked by DNS hierarchies.

Issues : Volatile Resources : No guarantee on when peers are available, when or how long resources are available.

Client-Server model



- Consider a file of size F.

Minimum upload time $\frac{F}{u_s} \times N$ (N copies for N users)

Max download time $= \frac{F}{d_{min}}$ $= \min\{d_1, d_2, \ldots d_N\}$

The minimum distribution time in client-server mode

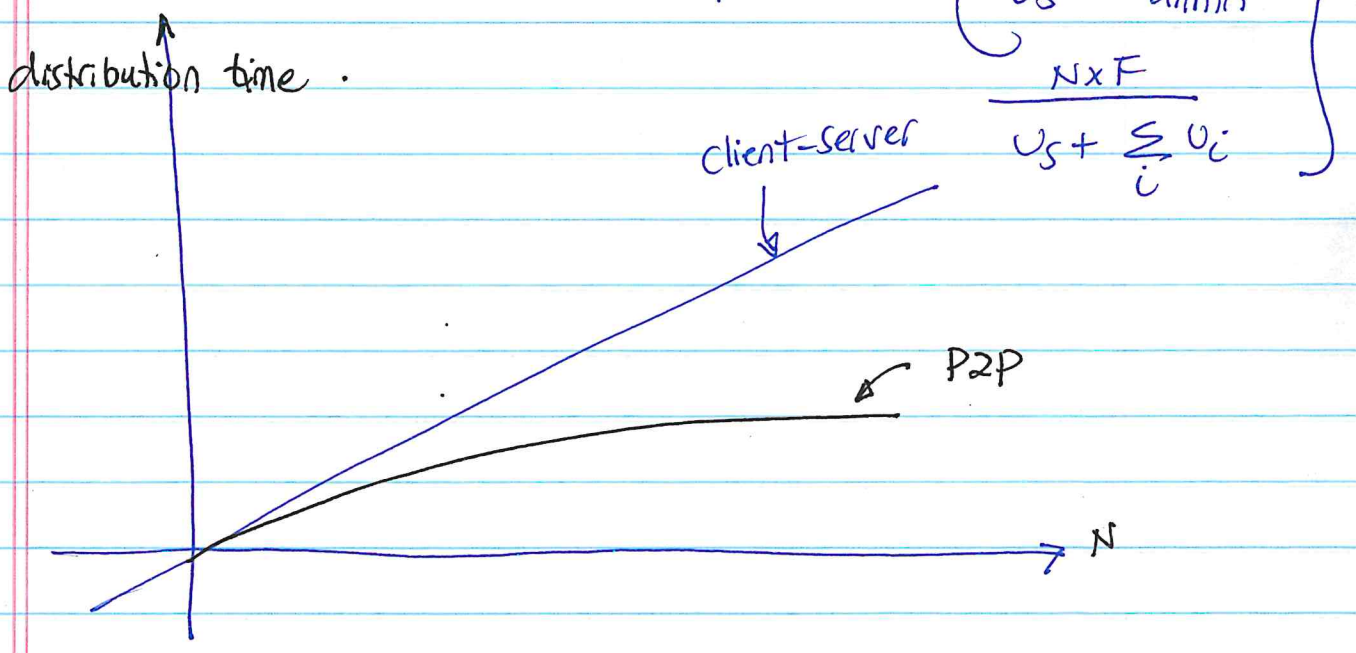
$$D_{cs} \geq \max\left\{\frac{F \cdot N}{u_s}, \frac{F}{d_{min}}\right\}.$$

Same File distribution in P2P systems

- The originator must upload one copy $= \frac{F}{u_s}$
- The slowest peer takes the longest time to download $= \frac{F}{d_{min}}$
- however, distributing F bits among N peers is now done through the combined upload rate

$$= \frac{N \times F}{u_s + \sum_i u_i}$$

Total distribution time $D_{P2P} \geq \max \left\{ \frac{F}{u_s}, \frac{F}{d_{min}}, \frac{N \times F}{u_s + \sum_i u_i} \right\}$



P2P Analysis

- Needs on only upload one copy of F
  - min. upload time $= \frac{F}{u_s}$
- Max. download time $= \frac{F}{d_{min}}$

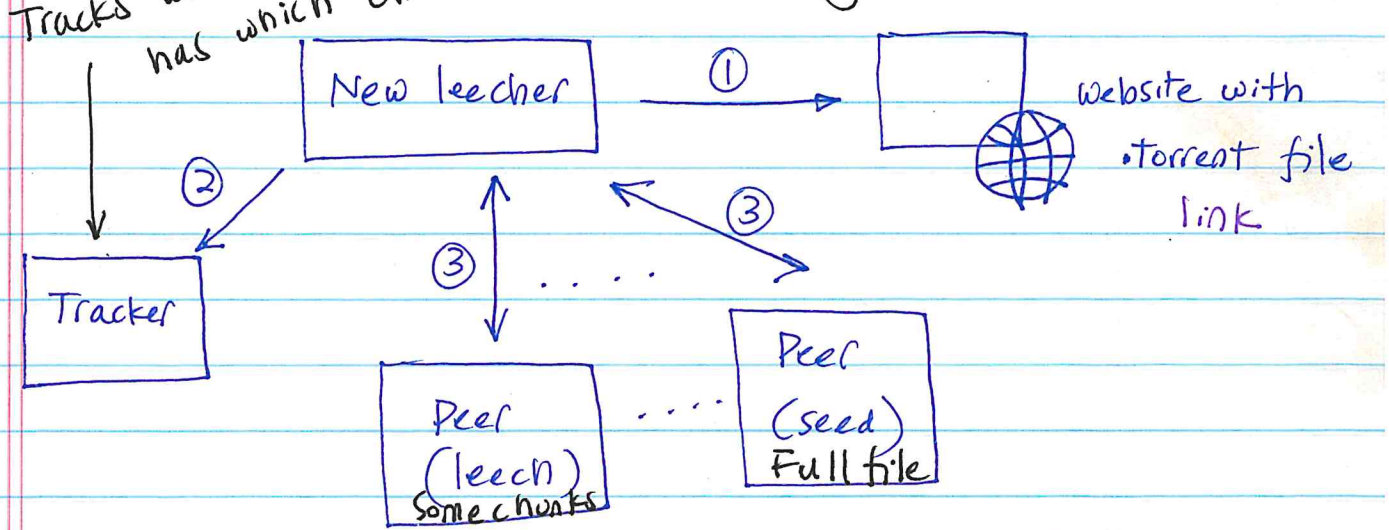
However, the combined upload rate $(u_s + u_1 + \cdots + u_N)$ is in effect when uploading multiple copies.

→ Time to deliver F bits to N users $= \frac{N \times F}{u_{Total}}$.

The maximum distribution time $= \max\left\{\frac{F}{u_s}, \frac{N \times F}{u_{Total}}, \frac{F}{d_{min}}\right\}$

Bittorrent

- Before Bittorrent
  - NAPSTER : A central DB for indexing
    - client-server search, direct P2P transfer.
  - KaZaA (installed) & Gnutella (self promoting)
    - Flooding based search
      - Performs well in finding popular items, poorly for unpopular items, poor guarantees on search quality.
      - Distributed but unstructured indexing



① Get .torrent file link from host website

Torrent: A collth of all peers participating in file distribution

.torrent link: Tracker (s) info (IP / Port)

② Contact tracker to get a list of peers.

③ Establish Concorrent TCP conn^{ns} with peers.

- Get a list of chunks from each peer
- Request for chunks not yet available.

- Bittorrent uses Local Rarest First priority downloading
  - prioritize least replicated copy downloading
  - Improve availability for everyone.

- Tit-for-tat Bandwidth usage.

Recent developments: Magnet links

- magnet has sufficient info about peers to download from them directly
- Theoretically can by pass Trackers.
- uses DHT.

DHT (Distributed Hash Tables)

- structured search, not centralized, flooding based
  - Distributed Indexing.
- provides guaranteed lookup successes, provable bounds on search times, and scalable.

- Key space distributed among peers. Each peer responsible for a portion of key space.

- Key-value pairs

Content
e.g. Arch 2019.01

Peer info.
xxx.xxx.xxx.xxx : yy.

chord

other examples: CAN, pastry, Tapastry

- Data items and nodes all identified by m-bits (limited by hash size).

- Address (key) space arranged in a circle (ring)
  size of 2^m
  e.g. SHA-256

  Peer ID: SHA-256 (IP:port) = N
  Key ID: SHA-256 (data_to_store) = K
  - A flat address space for both nodes and objects

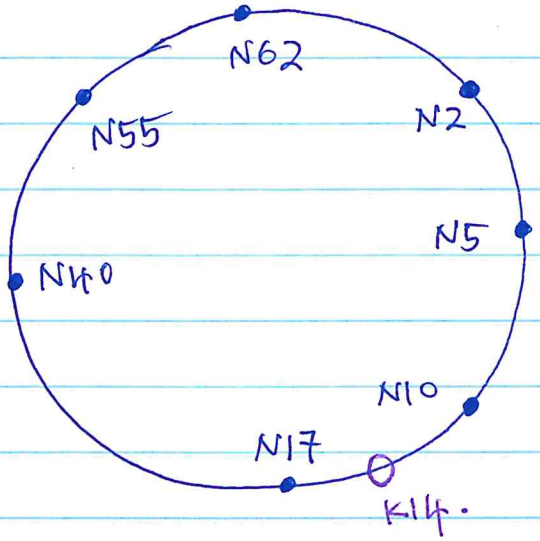
- Each peer has a finger table to resolve queries for keys.
  - Acts as a routing table.

  - each peer knows about m successors and one predecessor
  - Given a key K, the finger table tells which peer is responsible for it
    - peer might hold key or
    - peer might have info on who has the key.

- i^{th} entry of peer n is the first peer with

$$N \geq (n + 2^i) \bmod 2^m$$

Example chord with $m=6$.



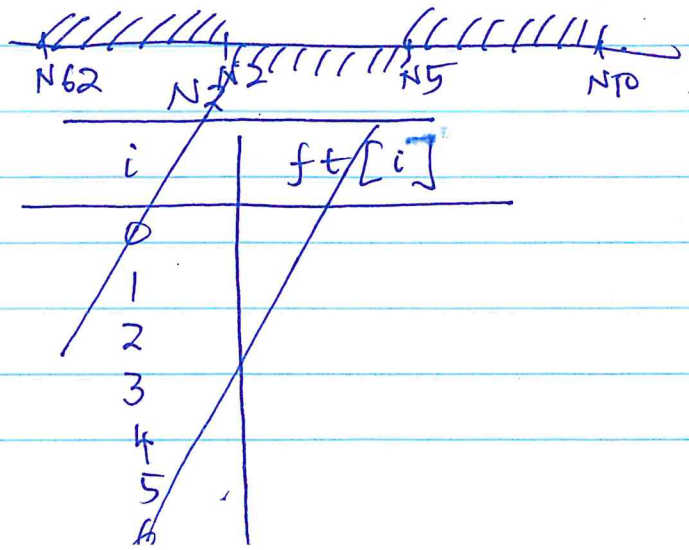
seven

- ~~Six~~ peers (nodes)
- N2 wants to share file [A]

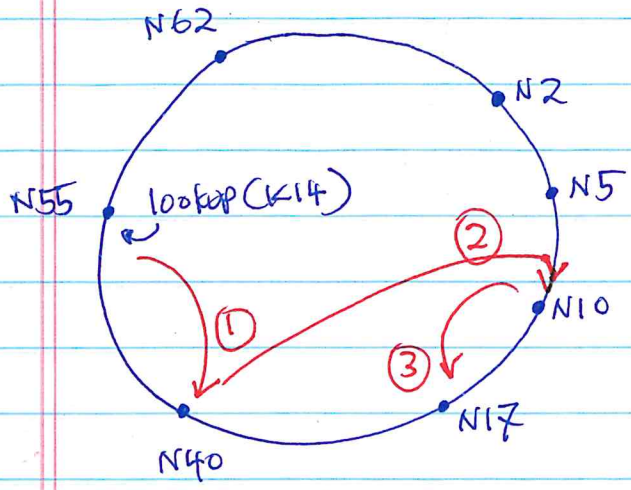
- Assume Hash(A) = 14

- Smallest peer with ID ≥ key ID in this case is N17.
  → K14 is N17's responsibility.

Finger Tables



- Suppose N55 is issuing a lookup for K14.
  - lookup(K14).



N55

i	FT[i]
0	N62
1	N62
2	N62
3	N2
4	N10
5	N40 ①
pre	N40

N40

i	FT[i]
0	N55
1	N55
2	N55
3	N55
4	N62
5	N10 ②
pre	N17

N10

i	FT[i]
0	N17 ③
1	N17
2	N17
3	N40
4	N40
5	N55
pre	N5