50.005 CSE

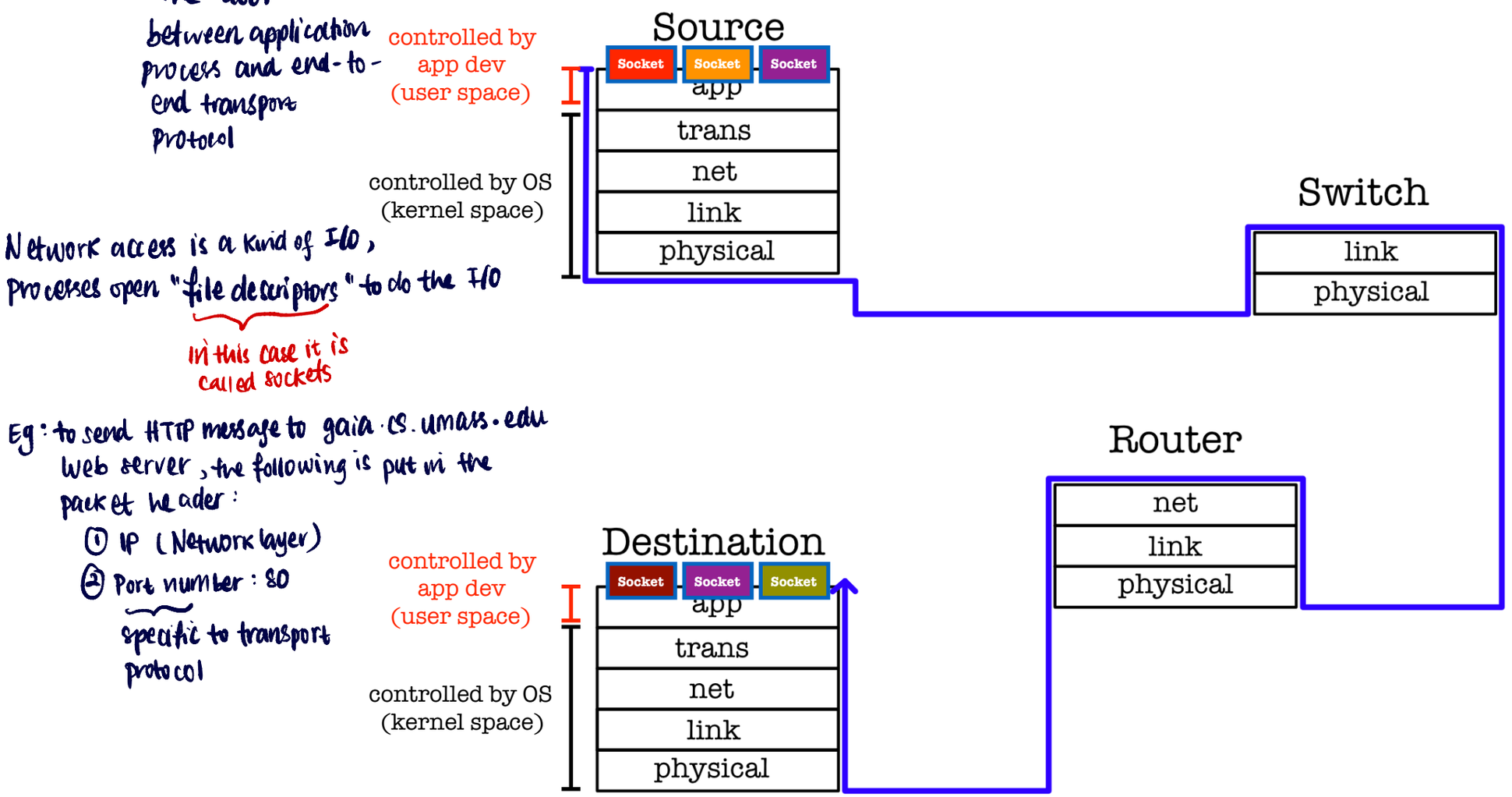
Natalie Agus
Information Systems Technology and Design
SUTD

Recap: DNS helps to resolve domain names (web address, human readable) into IP addresses (machine readable) - *so that you can be found in the internet.*

Issue: Your machine runs several programs to the same time. We need a way to pass the packet from the internet to the application that needs it, e.g: Steam, Web Browser, Telegram, etc.

Solution: SOCKET - a combination of IP plus port

The "door" between application process and end-to-end transport protocol



Network access is a kind of I/O, processes open "file descriptors" to do the I/O

In this case it is called sockets

Eg: to send HTTP message to gaia.cs.umass.edu web server, the following is put in the packet header:

1. IP (Network layer)
2. Port number: 80

specific to transport protocol

MULTIPLEXING

at sender, handling data from multiple sockets to **send out**.
Add transport packet header (for the layer below app layer)

Each app can listen to a specific port number

16 logical representation

Reserved port # :

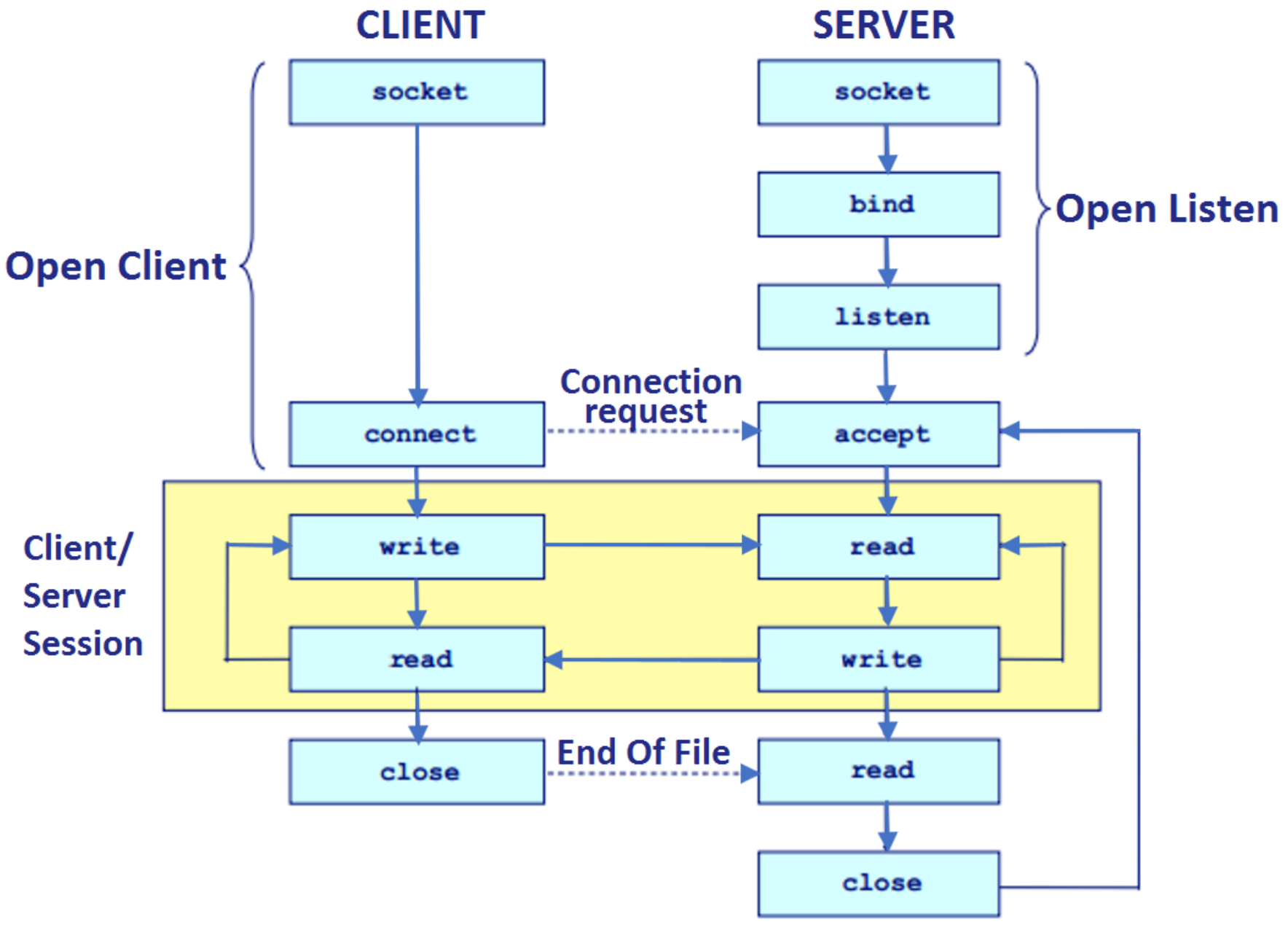
0 - 1023 : Well known ports allocated to SERVER services. For example, web servers use port 80, SMTP servers use port 25

1024 - 49151 : registered ports, can be registered for services with the IANA

49151 - 65535 : free to use by client programs

DEMULTIPLEXING

at receiver, handling data from multiple sockets to **receive in**.
Use transport packet header to deliver to the right socket number

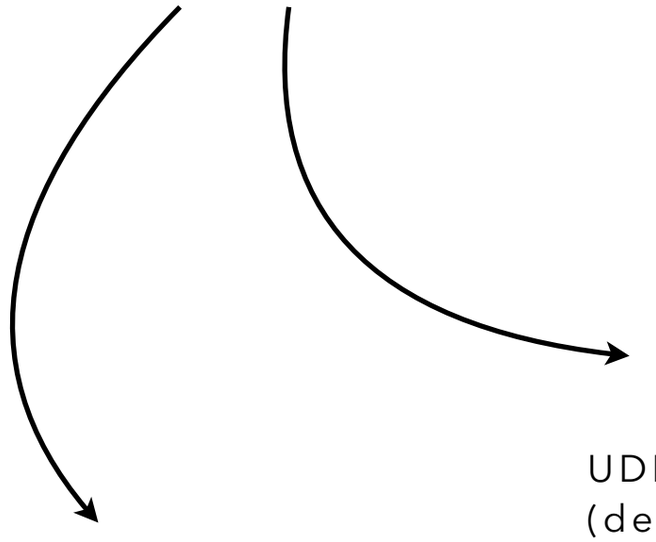

CLIENT
SERVER
socket
socket
bind
listen
Open Client
Open Listen
Connection request
connect
accept
write
read
read
write
Client/ Server Session
close
End Of File
read
close

SOCKET API

COMMUNICATION PROTOCOLS

DNS is used to translate between web address to IP address.
Then, we use socket to direct the packet to the right app in the computer.

Now we need **communication protocols** between two applications over the internet.



UDP

UDP socket identified by 2**-tuple**:
(destination IP, destination port #)

TCP

TCP socket identified by **4-tuple**:
(source IP, source port #, destination IP, destination port #)

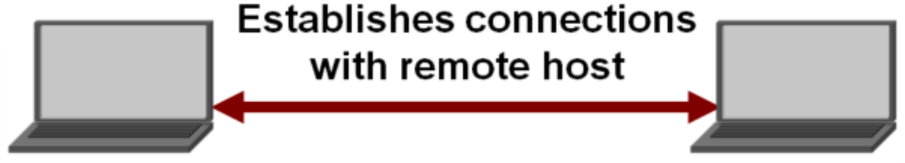
Socket programming in **application layer** relies on **TCP and UDP** services in **transport layer**

5 Application Layer

The Application layer is the group of applications requiring network communications.

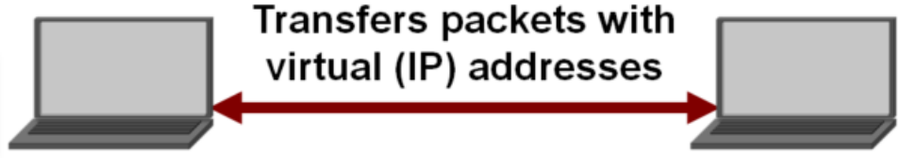
4 Transport Layer (TCP/UDP)

The Transport layer establishes the connection between applications on different hosts.



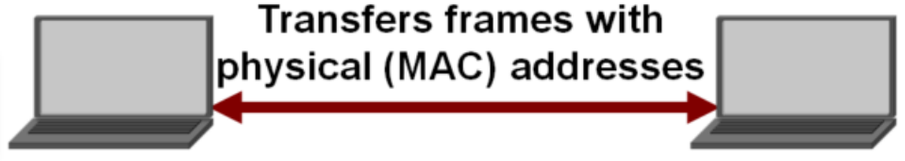
3 Network Layer (IP)

The Network layer is responsible for creating the packets that move across the network.



2 Data Link Layer (MAC)

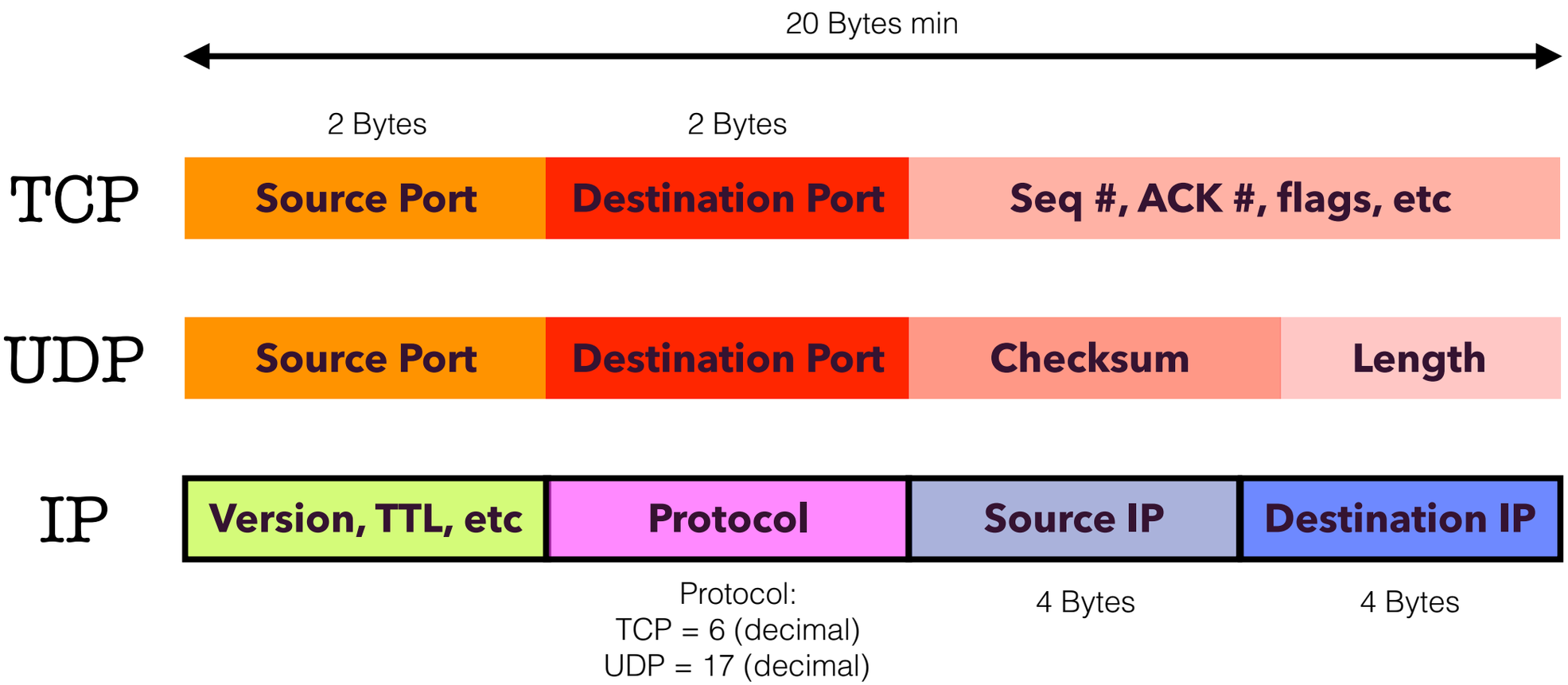
The Data Link layer is responsible for creating the frames that move across the network.



1 Physical Layer

The Physical layer is the transceiver that drives the signals on the network.

PACKET HEADERS

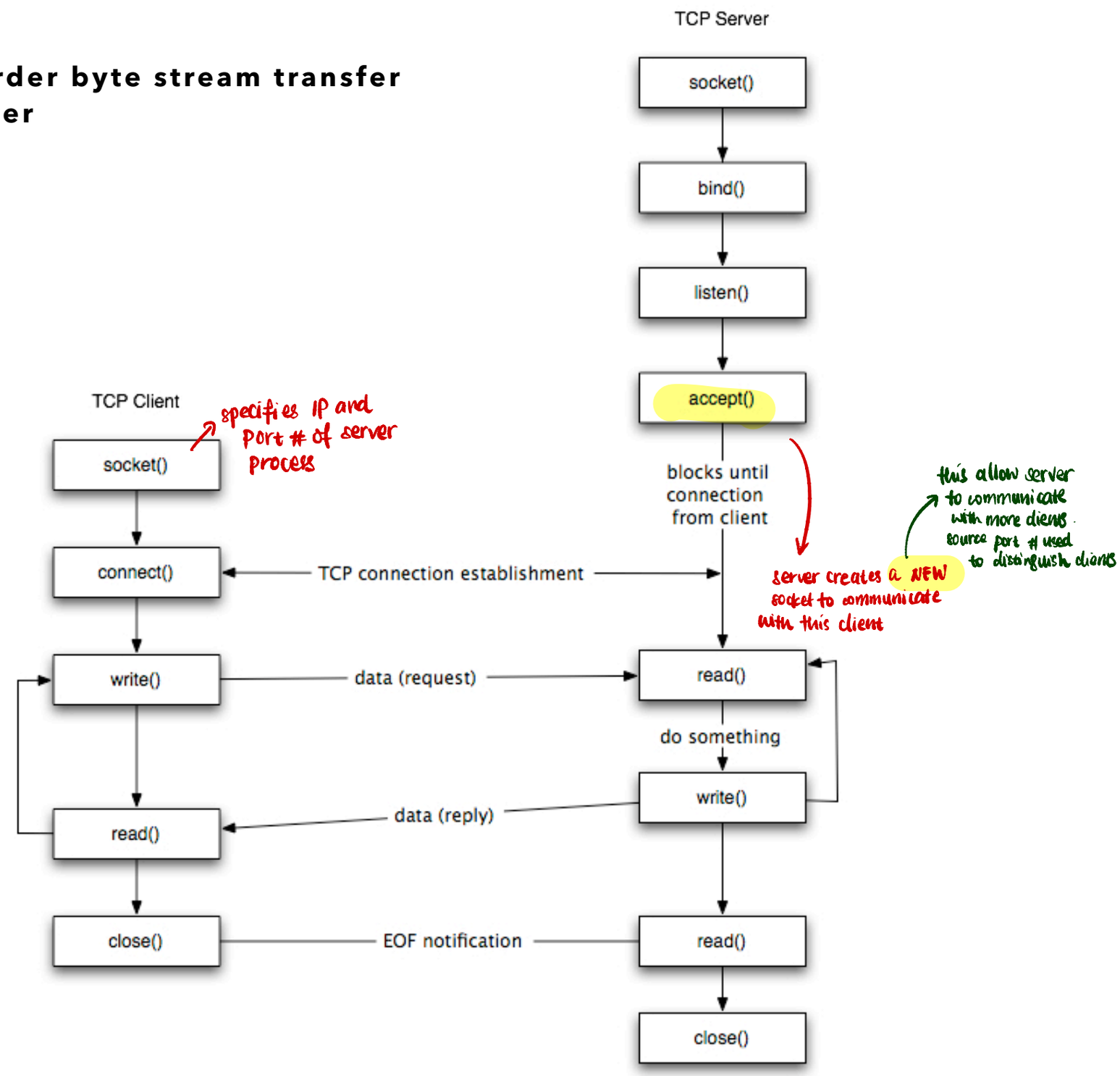


EXAMPLE PORT NUMBERS + SERVICE

Service	Detail	UDP / TCP port	App using
DNS	Domain name - IP resolution	UDP #53	—
DNS	Domain name - IP resolution TCP	TCP #53	—
HTTP	Web	TCP #80	World Wide Web, FaceTime, iMessage, iCloud, QuickTime Installer, Maps, iTunes U, Apple Music, iTunes Store
HTTPS	Secure Web SSL	TCP #443	
SMTP	Simple mail transport	TCP #25	Mail (sending email); iCloud Mail (sending email)
TELNET	Telnet terminal	TCP #23	—
FTP	File transfer protocol (FileZilla, IE)	TCP #20, #21	
SSH	Secure shell (Terminal)	TCP #22	Xcode Server (hosted and remote Git+SSH; remote SVN+SSH)
AFP IP	Apple File Protocol/IP	TCP #447, #548	AppleShare, Personal File Sharing, Apple File Service

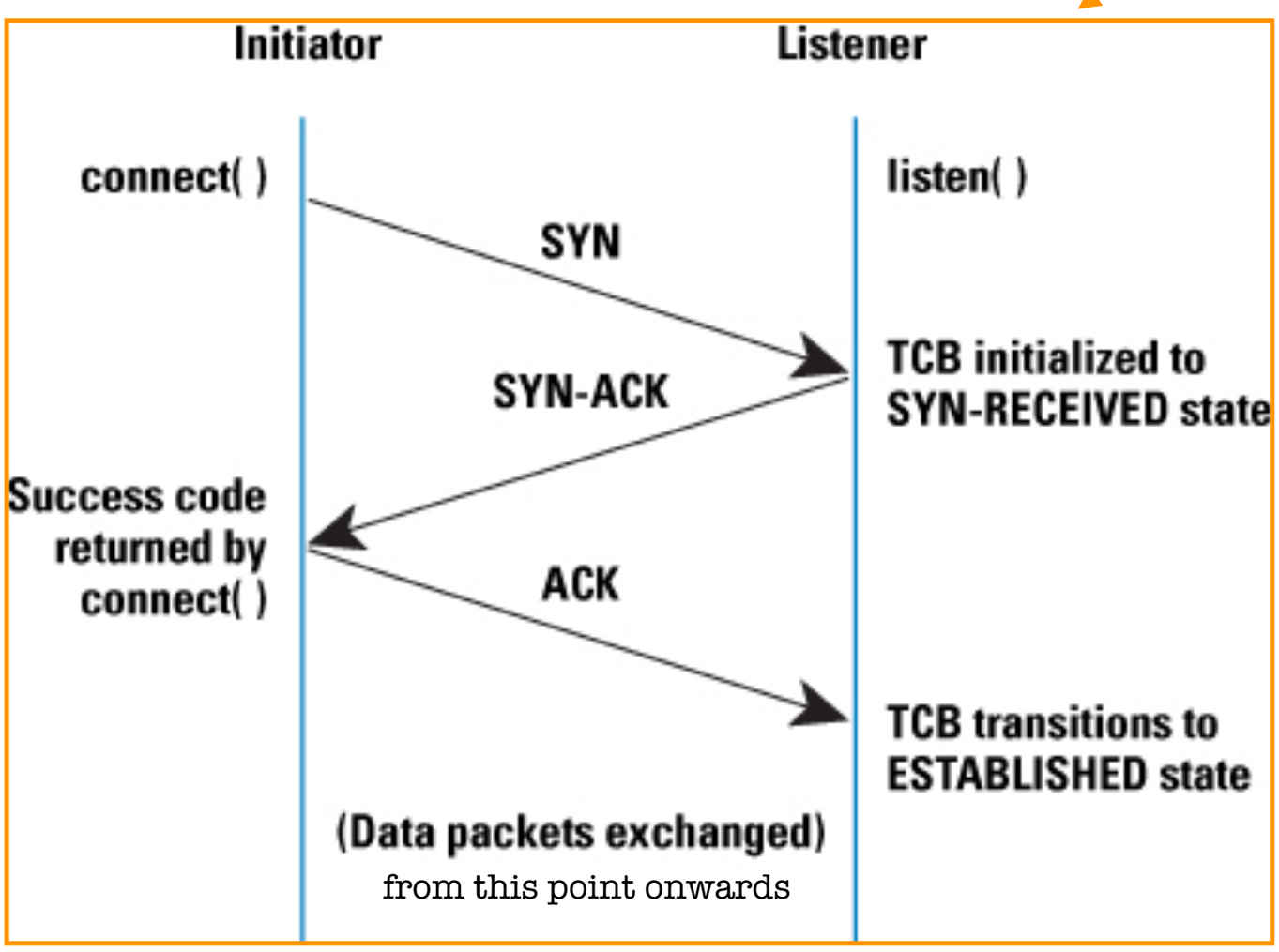
TCP

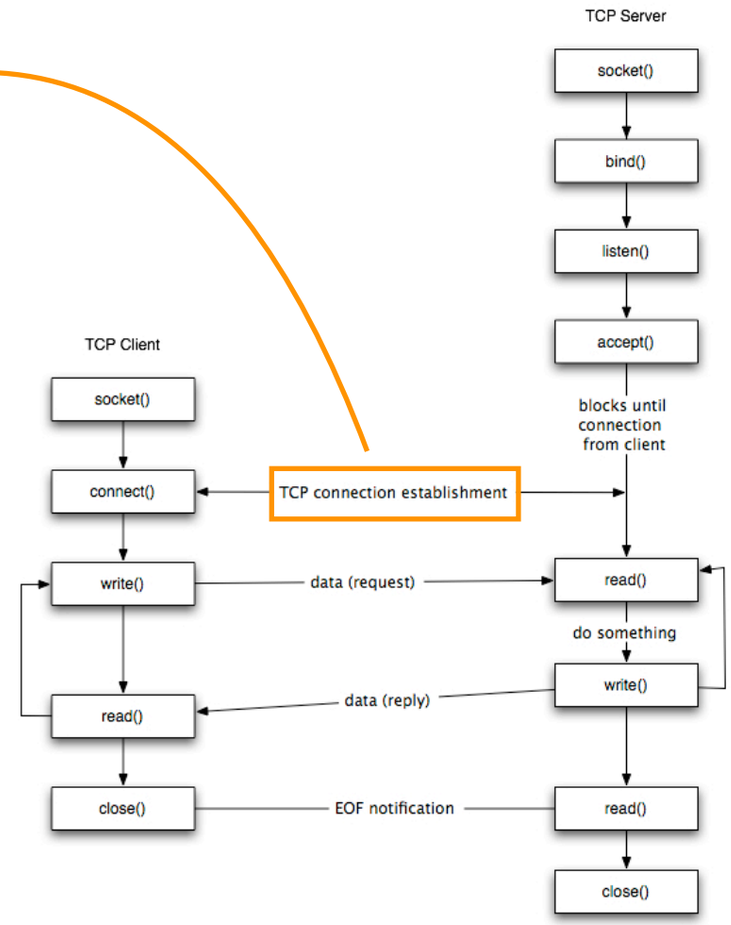
This protocol provides a **reliable, in-order byte stream transfer (called pipe) between client and server**



TCP CONNECTION ESTABLISHMENT

Its also known as **handshake. Its a 3-step protocol.**

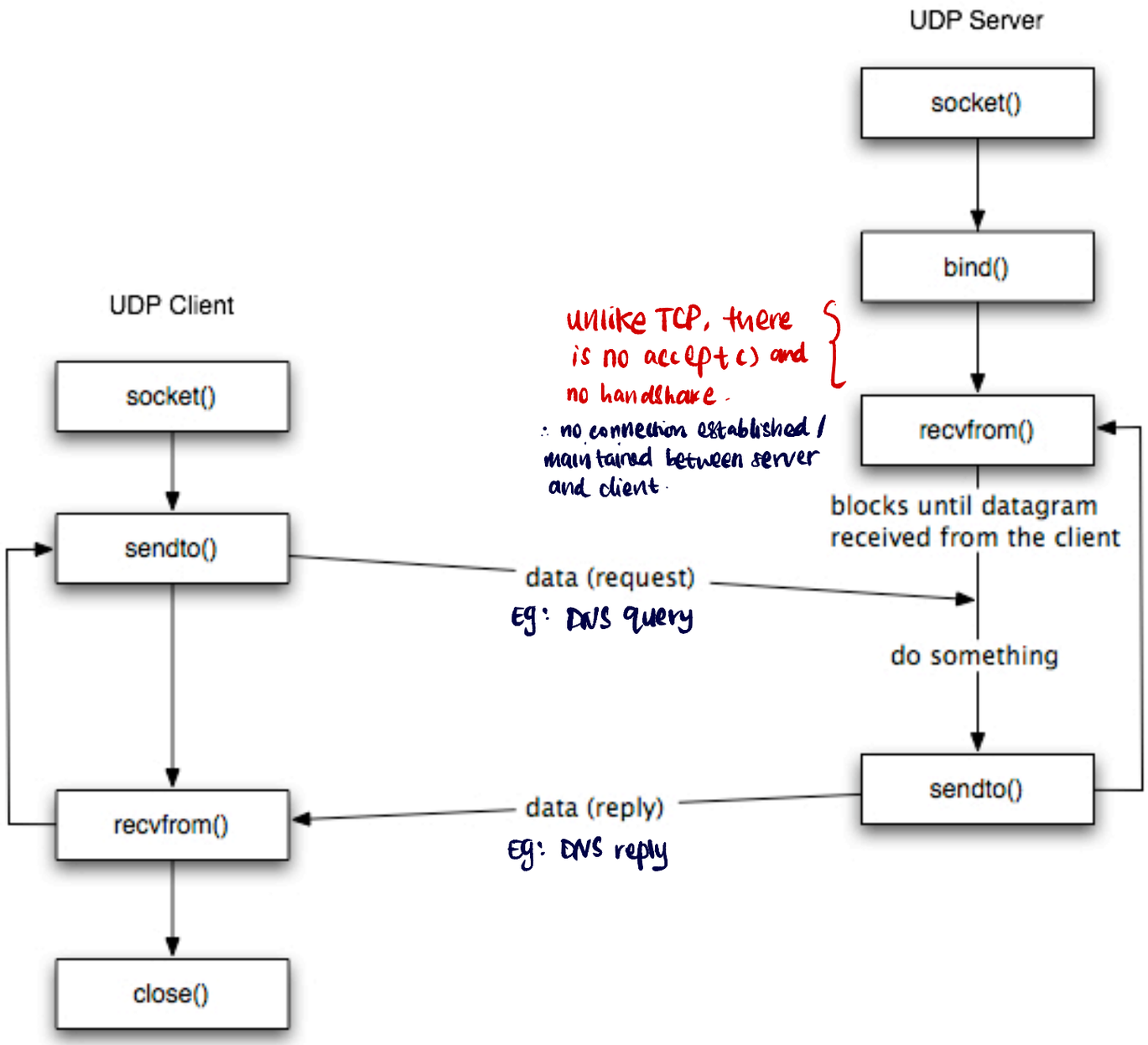




U D P

Sender explicitly attaches IP and port # of destination
receiver extracts these infos from the packet and pass to the correct app
Data can be lost along the way. No connection established between sender and receiver.

This protocol provides a **unreliable-but-fast, may-be-out-of-order transfer of bytes between client and server**



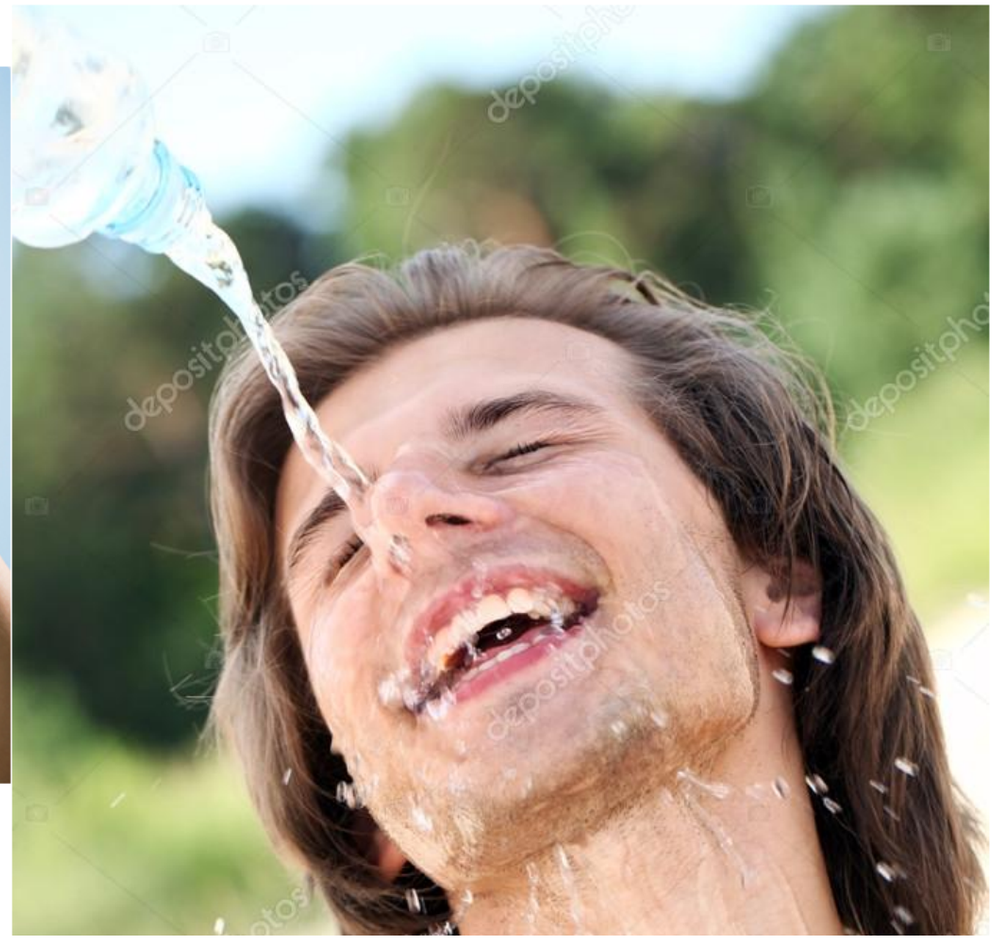
unlike TCP, there is no accept() and no handshake.
∴ no connection established / maintained between server and client.

Eg: DNS query

Eg: DNS reply

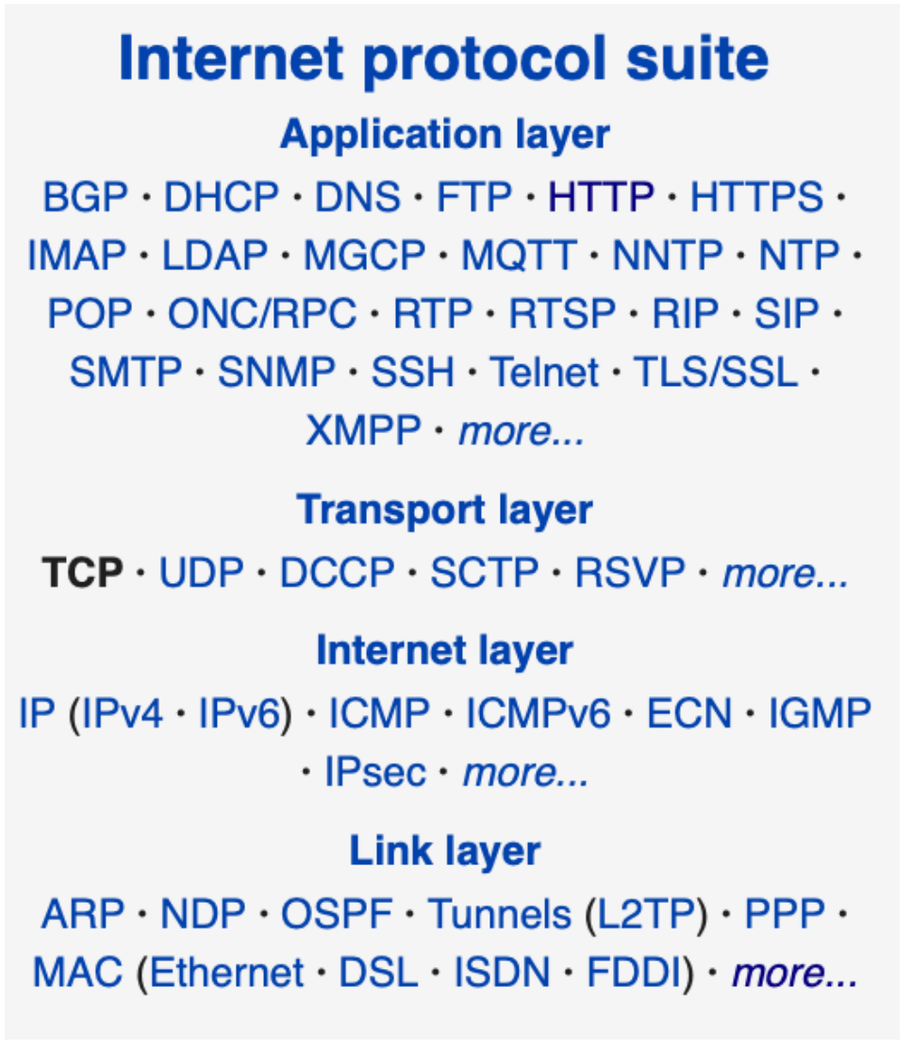
TCP

UDP



WHAT USES TCP: THE HTTP

hypertext transfer protocol

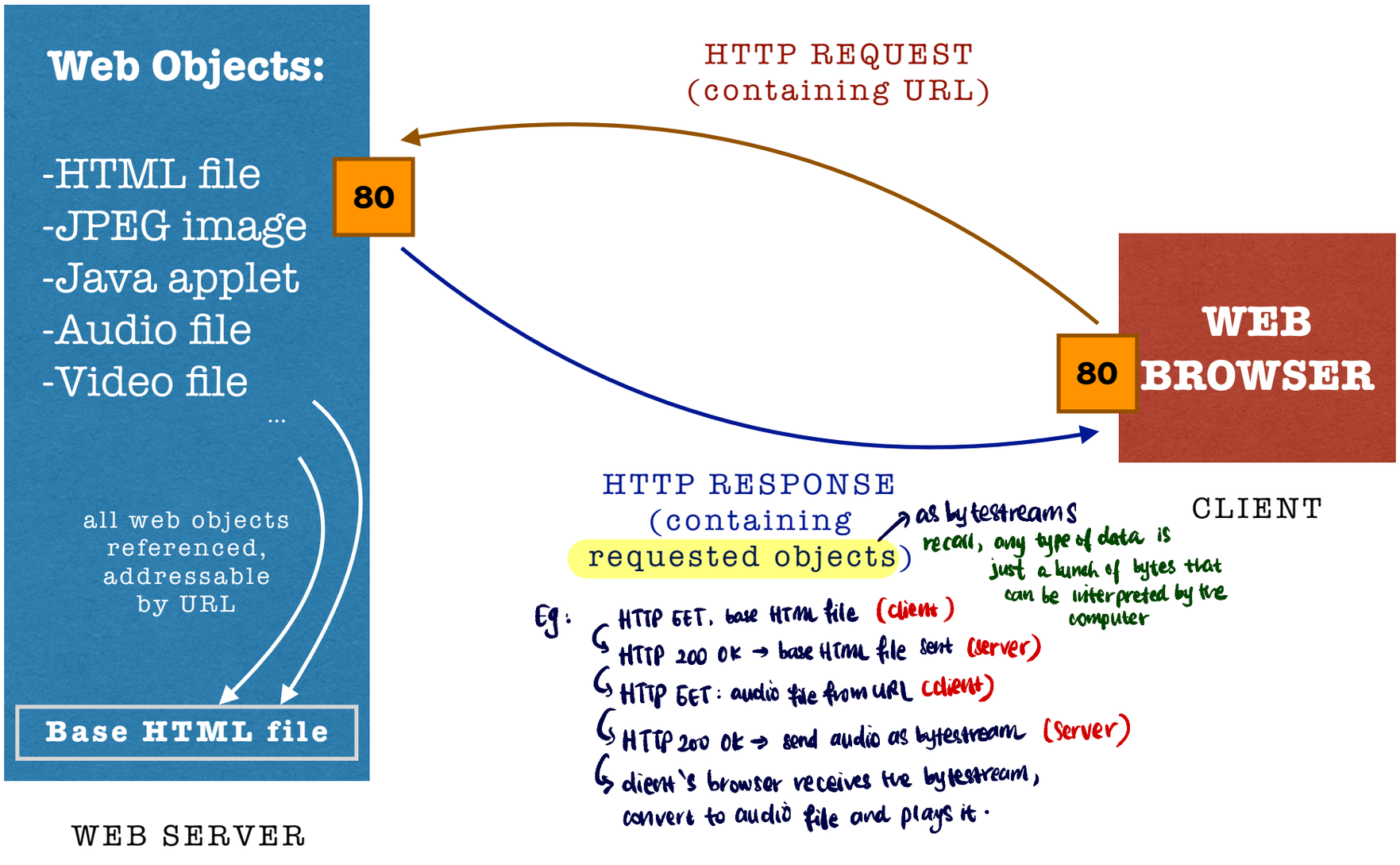


HTTP: Web's application layer protocol

It is a **client-server** model:

- Client = browser that **requests,** and process replies by displaying web objects
- Server = **sends** responds based on requests

A LITTLE INFO ON HOW THE WEB WORKS....



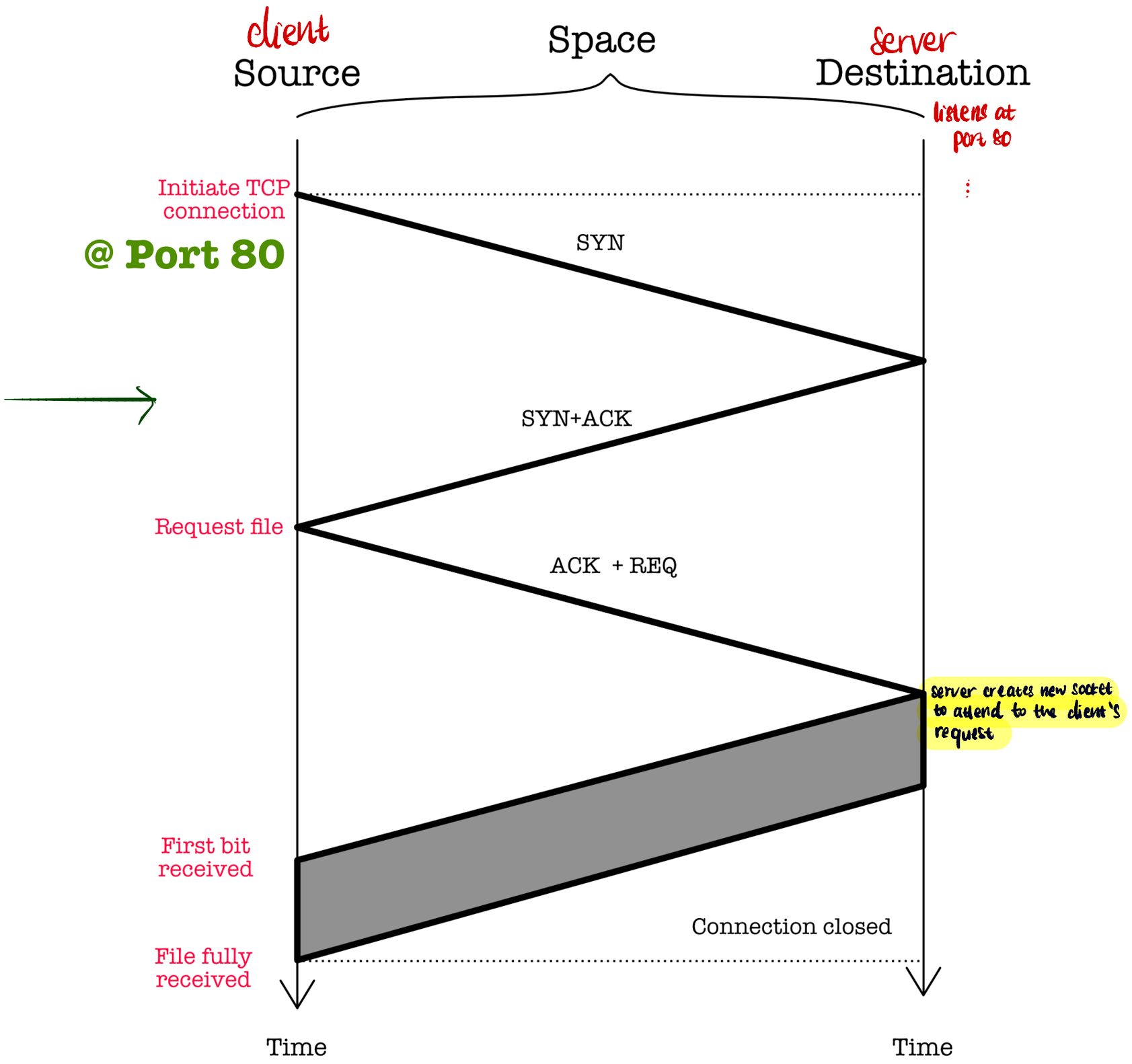
HTTP

HTTP (app layer protocol) uses TCP (net layer protocol).

Basic HTTP is **stateless**, server doesn't have past information on client's requests. * cookies can be added to keep state

Two types of connection:

1. **Non-Persistent HTTP (HTTP 1.0)**
2. **Persistent HTTP (HTTP 1.1)**



HTTP

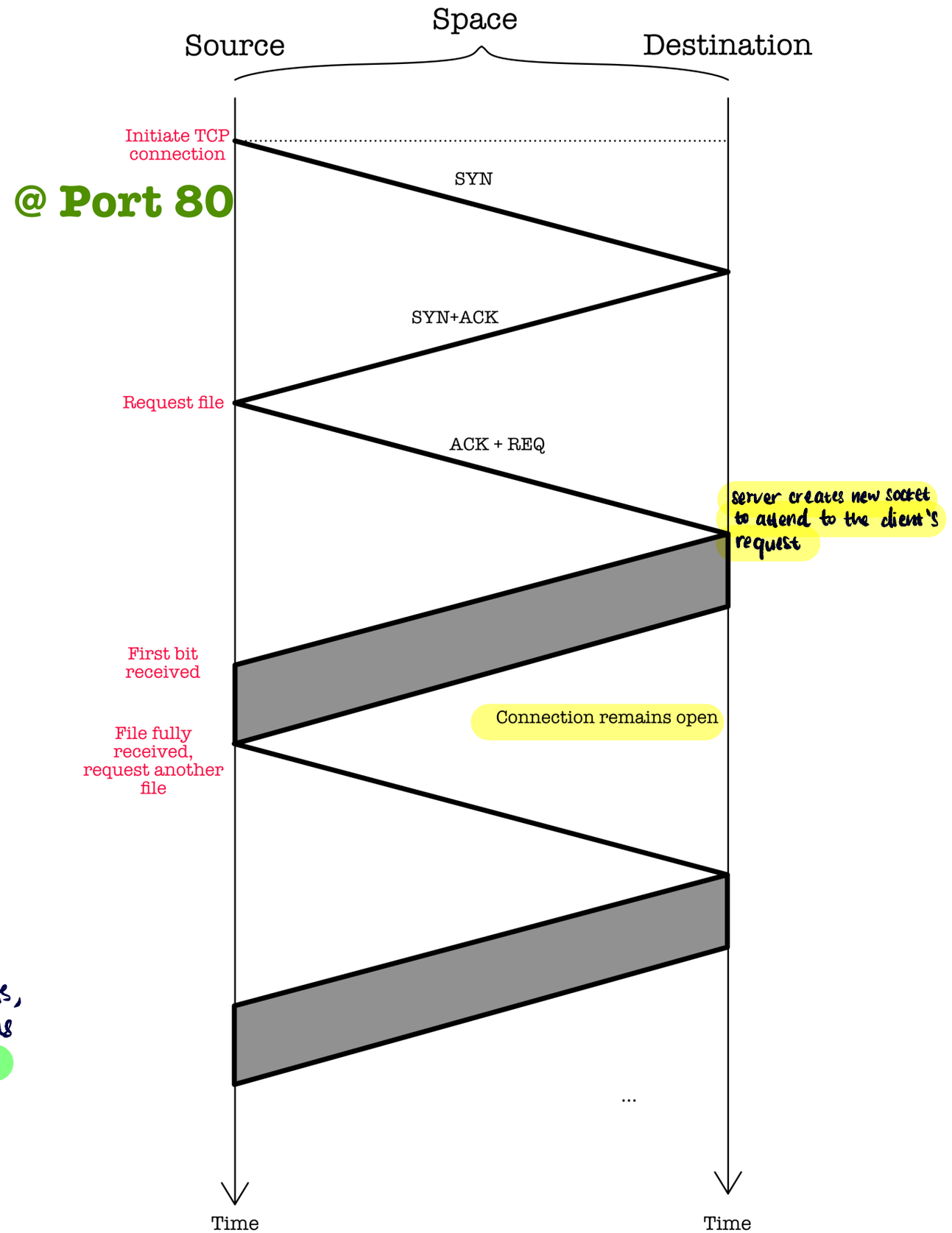
HTTP (app layer protocol) uses TCP (net layer protocol).

Basic HTTP is **stateless,** server doesn't have past information on client's requests.

Two types of connection:

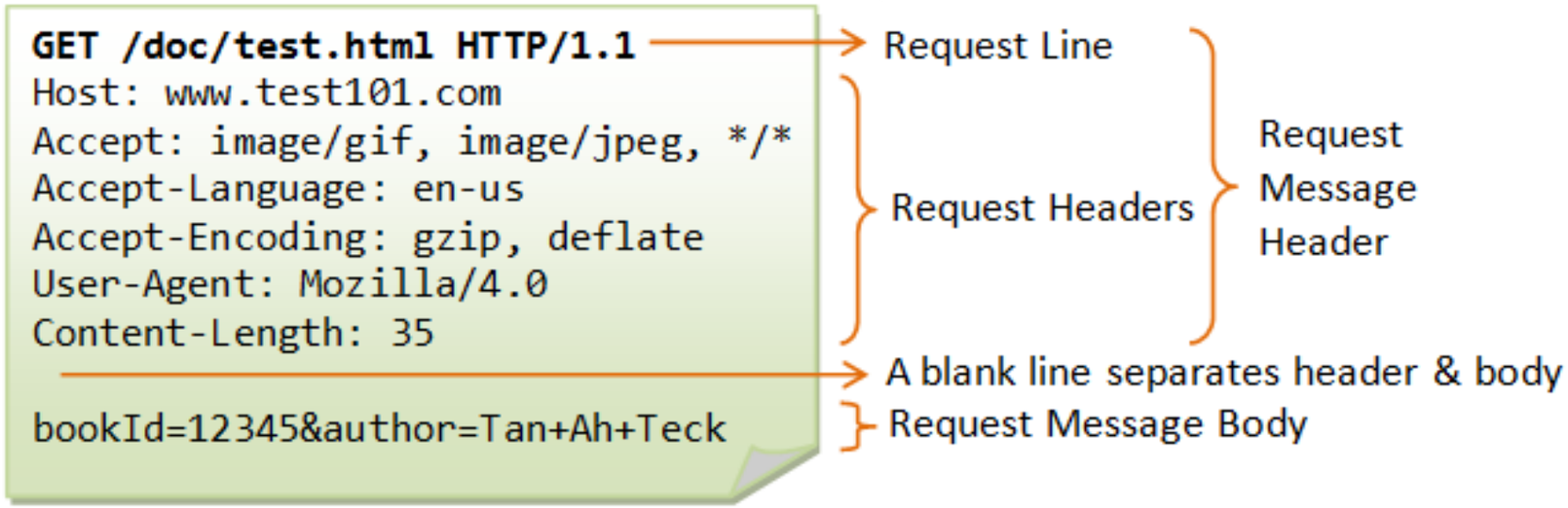
1. **Non-Persistent HTTP (HTTP 1.0)**
2. **Persistent HTTP (HTTP 1.1)**

1.0	VS	1.1
- one connection per object request - Requires 2RTT per object - OS overhead for each TCP connection - Browsers often open parallel TCP connections to fetch referenced objects to increase link utilization		- connection remains open at server after sending response - 1RTT for all referenced objects - 2RTT for the base HTML file (the first thing to fetch) - Pipelining is allowed: send more requests as soon as client encounters referenced objects, no need to wait for previous request to finish to increase link utilization



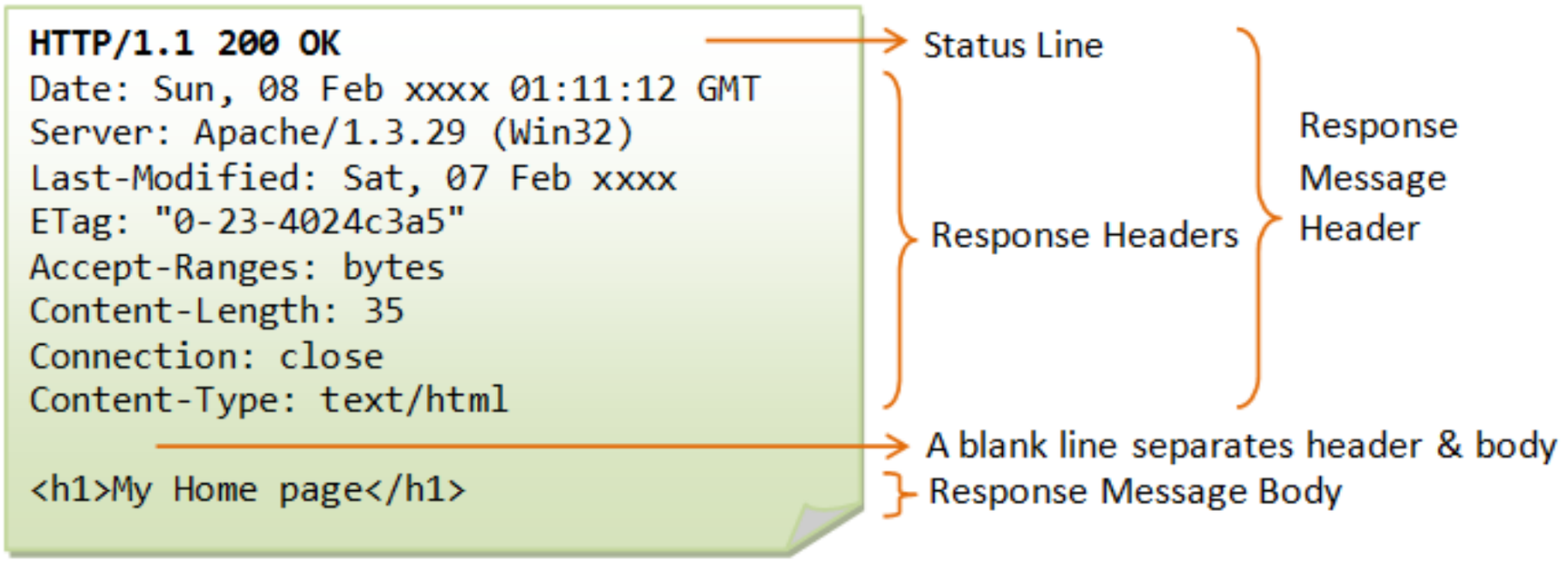
HTTP REQUEST MESSAGE

(GET/POST/HEAD: HTTP 1.0)
(GET/POST/HEAD/PUT/DELETE : HTTP 1.1)



HEAD: asks server to leave requested object out of response
PUT: uploads file in entity body to path specified in URL field
DELETE: deletes file specified in the URL field

HTTP RESPONSE MESSAGE



200 : OK
301 : moved permanently, new location specified at body
400 : Bad request (not understood)
404 : not found
505 : HTTP version not supported

HTTPS

Uses SSL (Secure Socket Layer) encryption. Runs at PORT 443.

Most websites run on HTTPS now (that lock symbol on ur browser)

Why use HTTPs?

- Intruders both malignant and benign exploit every unprotected resource (cross-site scripting) between your websites and users: **they inject (contents like ads), intercept requests, redirects websites**
- Many intruders look at aggregate behaviours to identify your users: **people can learn who uses your websites and listen to sensitive information like login credentials**
- HTTPS doesn't just block misuse of your website. It's also a requirement for many cutting-edge features and an enabling technology for app-like capabilities such as service workers.

https://css-tricks.com/forums/topic/how-to-make-a-website-do-the-harlem-shake/

What cookies can be used for:

- authorization
- shopping carts
- recommendations
- user session state (Web /email)
- personalization

cookies permit sites to learn alot about you, eg: your name and email, search history on the website, etc

COOKIES

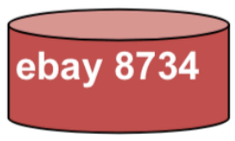
↗ text form

↓ to retrieve user-server state, made of 4 components:

1. cookie header line imbedded in HTTP response message
2. cookie header line imbedded in next HTTP REQ message
3. cookie file kept on user host, managed by browser
4. back-end database at the server

client

server



cookie file

usual http request msg

Amazon server creates ID 1678 for user

create entry

backend database 4

usual http response **set-cookie: 1678** 1

usual http request msg **cookie: 1678** 2

cookie-specific action

access

usual http response msg

note that websites can only read its own cookies for security reasons and are not shared among browsers

one week later:

usual http request msg **cookie: 1678**

cookie-specific action

access

usual http response msg

ads that follow you across websites are owned by the SAME advertising companies.

Eg: your shopping cart maintained with your past delivery info although you are not logged in.

other example:

you login to Facebook, HTTP POST with password + id from a new browser

↳ server verifies credentials, and send a HTTP response + set cookie header

↳ browser stores this cookie on disk. any subsequent navigation (HTTP req) will include this cookie.

↳ FB server reads cookie and store in some data structure

↳ next time the same client access FB, FB tailor the content for the user.

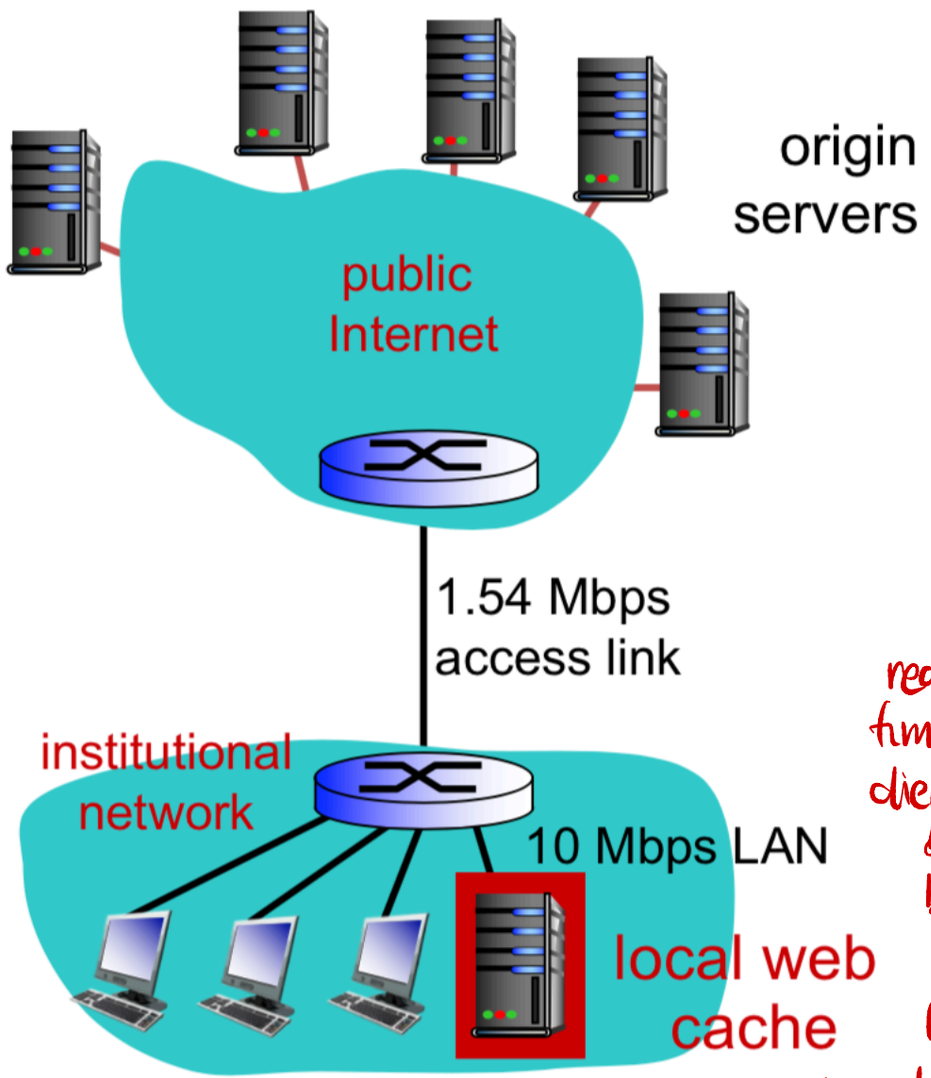
Eg: the client will see his unique activity feed

acts as both client and server:
① server to original requesting client
② client to original web server

WEB-CACHES

One of the reasons why it is faster to visit website the second time (DNS caching + Web content caching)

Done by: ISPs



Reduces access link utilization

If there's no local web cache, the access link utilization % ++, results in more queueing delay

Web caches is a **cheap solution** to **reduce** access link utilization (apart from increasing the access link speed itself, which is expensive)

and also reduce response time for client request because LAN bandwidth is generally faster (less traffic)

Modification to HTTP request:
CONDITIONAL GET

eg: