CS2105 Introduction to Computer Networks

Lecture 2
Application Layer

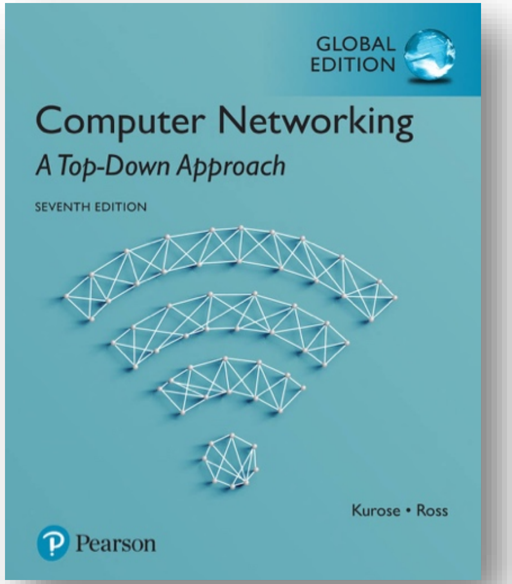
https://goo.gl/nZfUZ5

20 Aug 2018

Lecture 2 Quiz

https://goo.gl/nZfUZ5

Textbook



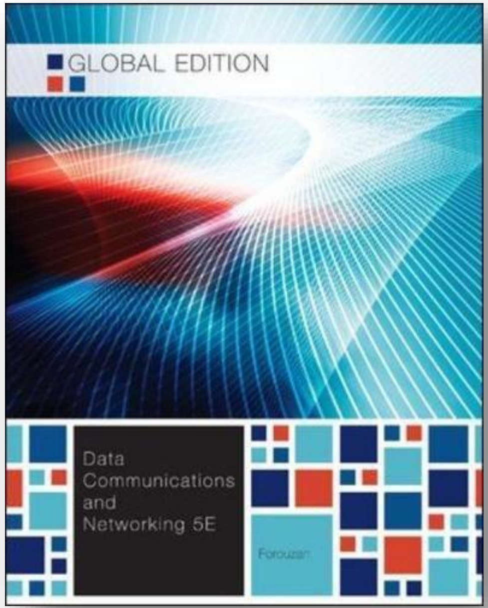
Application

Transport

Network

Link

Physical



Chapters 3-6

https://books.google.com.sg/books/about/Data_Communications_and_Networking.html?id=bwUNZvJbEeQC

Adi Yoga
Sidi Prabawa

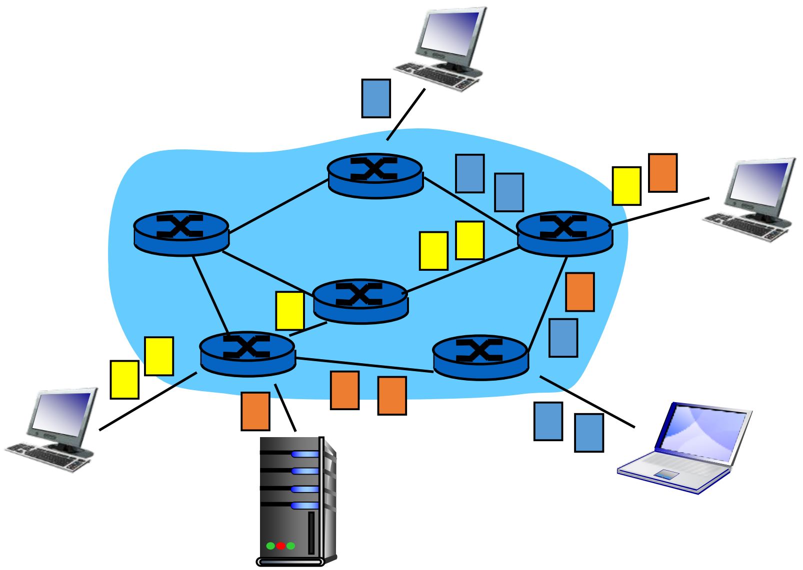
Previously…

Packet Switching

The Internet is a packet switching network

- Hosts share and contend network resources.
- Application message is broken into a bunch of packets and sent onto the link one by one.
- A router stores and forwards packets.
- Receiver assembles all the packets to restore the application message.

Bandwidth division into "pieces"
Dedicated allocation
Resource reservation

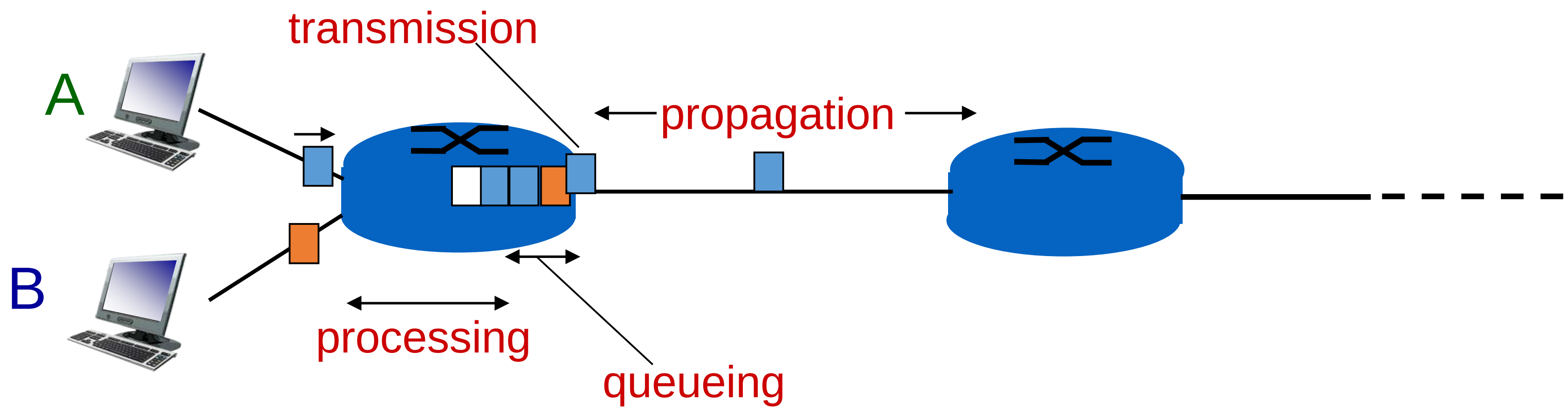


PREVIOUS LECTURE

Packet Delay

End-to-end delay is the time taken for a packet to travel from source to destination. It consists of:

- processing delay
- queueing delay
- transmission delay
- propagation delay



Network Protocols

PREVIOUS LECTURE

Networks are complex.

- many issues to consider
- support different applications, running on
- large number of hosts, through
- different access technology and physical media.

Protocols regulate communication activities in a network.

- Define the *format* and *order* of messages exchanged between hosts for a specific purpose.

Learning Outcomes

After this class, you are expected to:

- Know the basic HTTP interactions between the client and the server, including HTTP request (GET and header fields) and HTTP response.
- Know the concepts of persistent connection, parallel HTTP connections and stateless protocol.
- Know the services provided by DNS and how a query is resolved.

Application

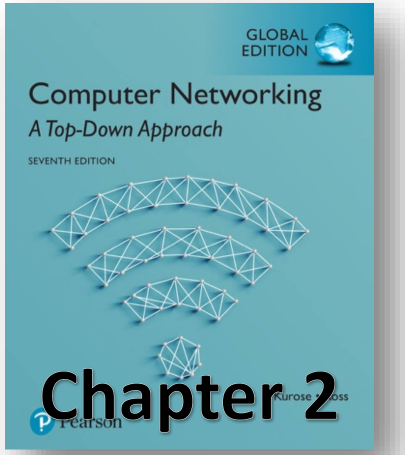
Transport

Network

Link

Physical

Lecture 2: Roadmap



2.1 Principles of Network Applications

2.2 Web and HTTP

2.5 DNS

2.7 Socket programming with TCP

2.8 Socket programming with UDP

Next week

*some slides taken from the publisher

Evolution of Network Applications

Early days of Internet

- Remote access (e.g. telnet, now ssh)

1970s – 80s

- Email, FTP

1990s

- World Wide Web
- Instant Messaging

2000s

- P2P file sharing
- Online games
- Skype, Facebook

2010 – now

- YouTube
- Web Apps
- Compute Cloud

The Application Layer Protocol is used by every Internet Application

Network applications run on hosts and contains communicating processes

Server Process

waits to be contacted

Client Process

initiates the connection

Application architecture

Possible structure of network applications

- client-server
- peer-to-peer
- hybrid

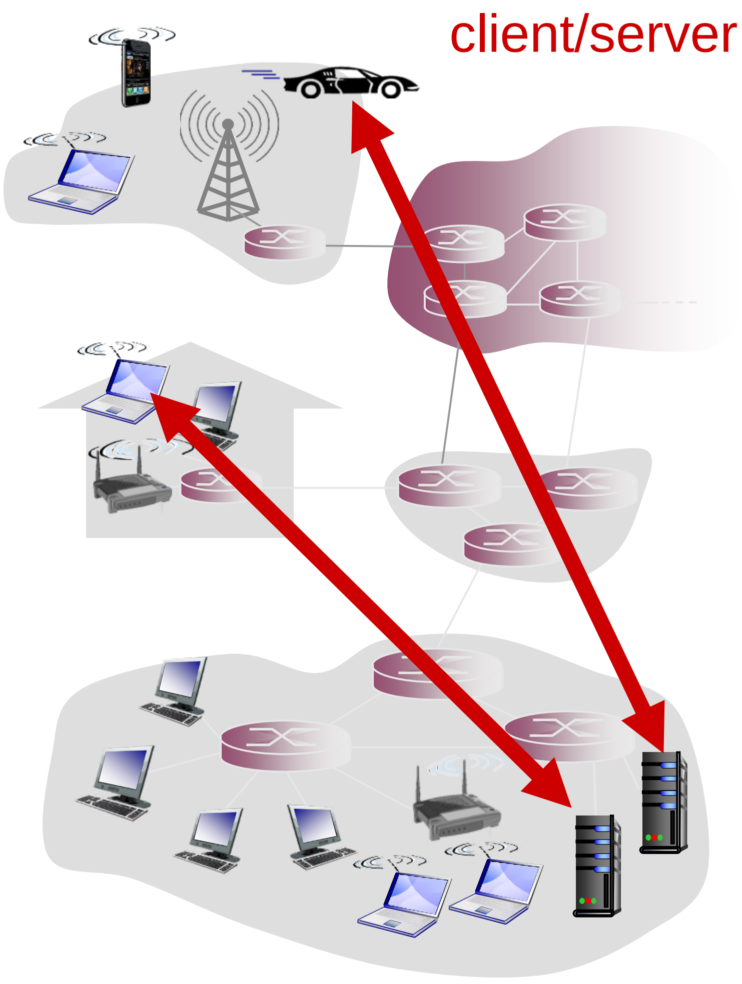
Client-Server Architecture

Server:

- Waits for incoming requests
- Provides requested service to client

Client:

- Initiates contact with server ("speaks first")
- Typically requests service from server
- For Web, client is usually implemented in browser

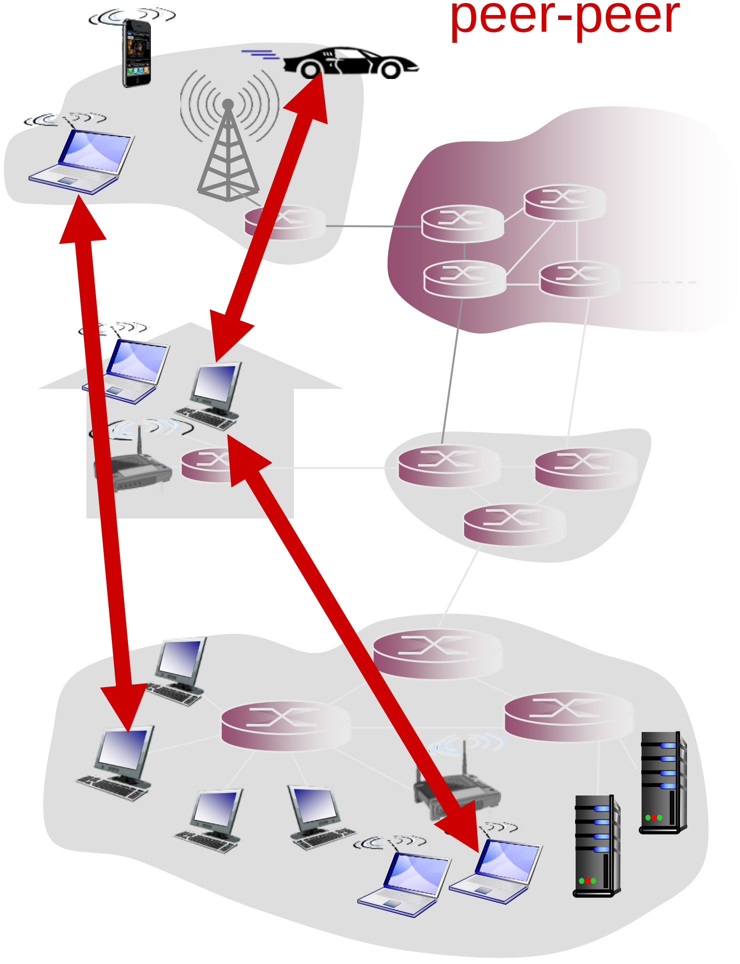


P2P Architecture

- No always-on server
- Arbitrary end systems directly communicate.
- Peers request service from other peers, provide service in return to other peers

Highly scalable

But difficult to manage



Hybrid of Client-Server and P2P

Example: instant messaging

- Chatting between two users is P2P
- Presence detection/location is centralized:
    - User registers its IP address with central server when it comes online
    - User contacts central server to find IP addresses of buddies

What transport service does an app need?

Data integrity

- some apps (e.g., file transfer, web transactions) require 100% reliable data transfer
- other apps (e.g., audio streaming) can tolerate some data loss

Timing

- some apps (e.g., online interactive games) require low delay to be “effective”

Throughput

- some apps (e.g., multimedia) require minimum amount of bandwidth to be “effective”
- other apps (e.g., file transfer) make use of whatever throughput available

Security

- encryption, data integrity, authentication …

Requirements of Example Apps

Application	Data loss	Throughput	Time-sensitive
File transfer	No loss	Elastic	No
Electronic mail	No loss	Elastic	No
Web documents	No loss	Elastic	No
Real-time audio/video	Loss-tolerant	Audio: 5kbps-1Mbps Video:10kbps-5Mbps	Yes: 100s of msec
Stored audio/video	Loss-tolerant	Same as above	Yes: few seconds
Interactive games	Loss-tolerant	Few kbps – 10 kbps	Yes: 100s of msec
Text messaging	No loss	Elastic	Yes and no

App-layer Protocols Define…

types of messages exchanged

- e.g., request, response

message syntax:

- what fields in messages & how fields are delineated

message semantics

- meaning of information in fields

rules

- for when and how applications send & respond to messages

open protocols:

- defined in RFCs
- allows for interoperability
- e.g., HTTP, SMTP

proprietary protocols:

- e.g., Skype

Rules for message exchange

- Depends on what service you need
- Depends on what service is provided

Sending snail mail

What services do you need?

- Tracking? Reliability? Receipt?

What service can delivery company provide?

- Registered mail? Insurance?

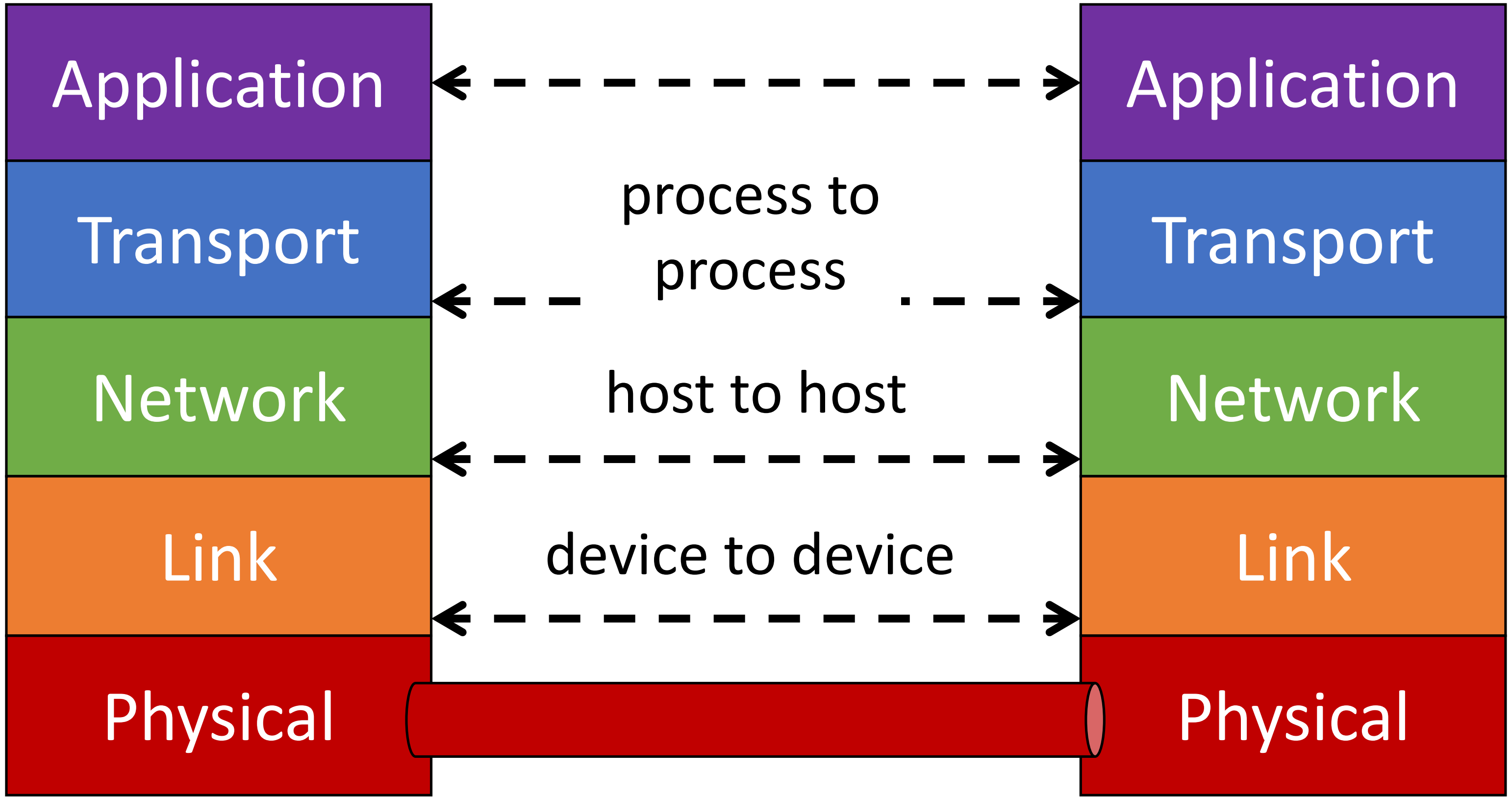
What if you need a service that delivery company cannot provide?

- Receive receipt with no registered mail?
- Tracking with no tracking service?

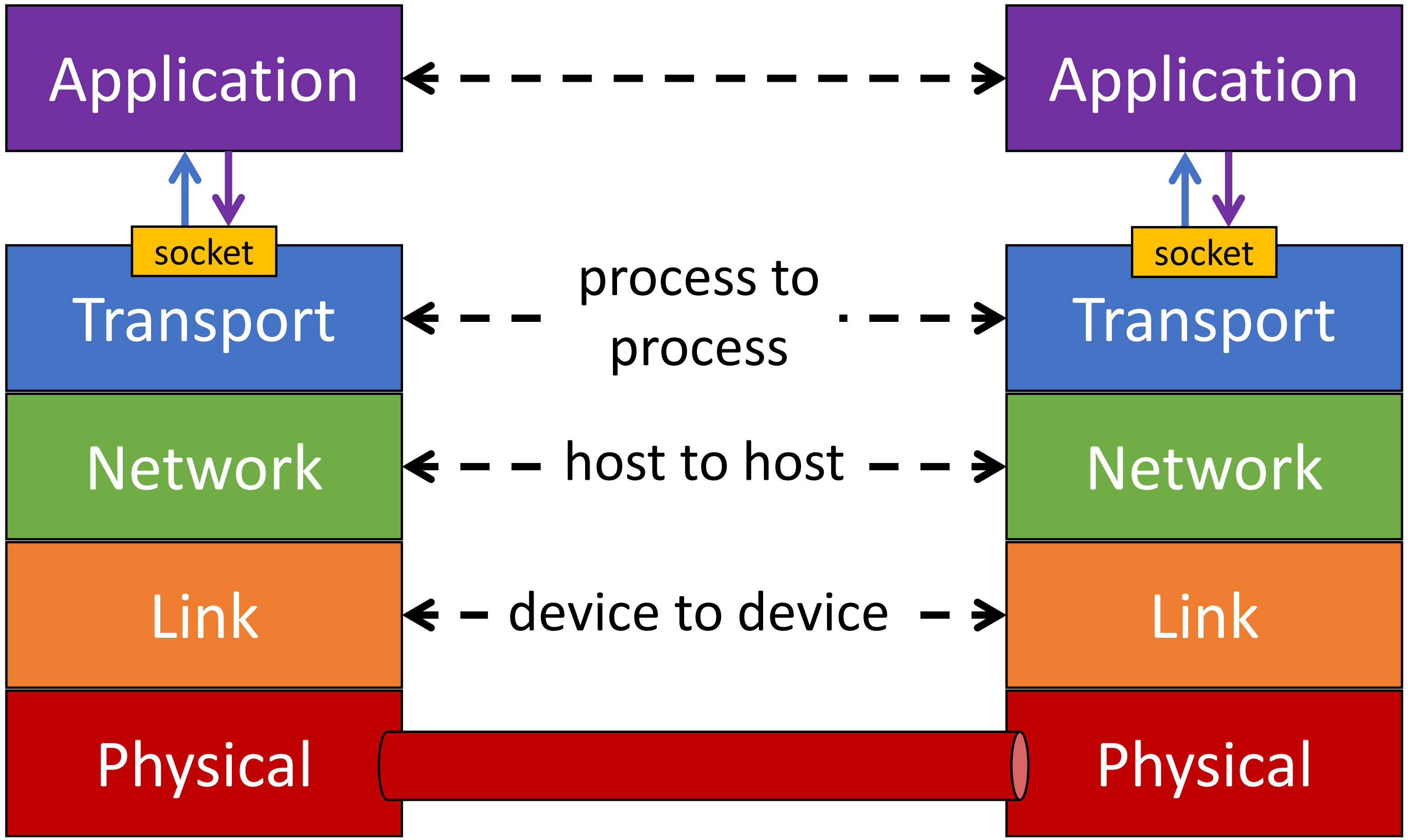
Build it into the Application Protocol

Imagine you are a web app

Who do you rely on to transmit your messages?



Sockets


Application
Application
socket
socket
Transport
process to process
Transport
Network
host to host
Network
Link
device to device
Link
Physical
Physical

Sending snail mail

Who do you rely on to deliver your letter?

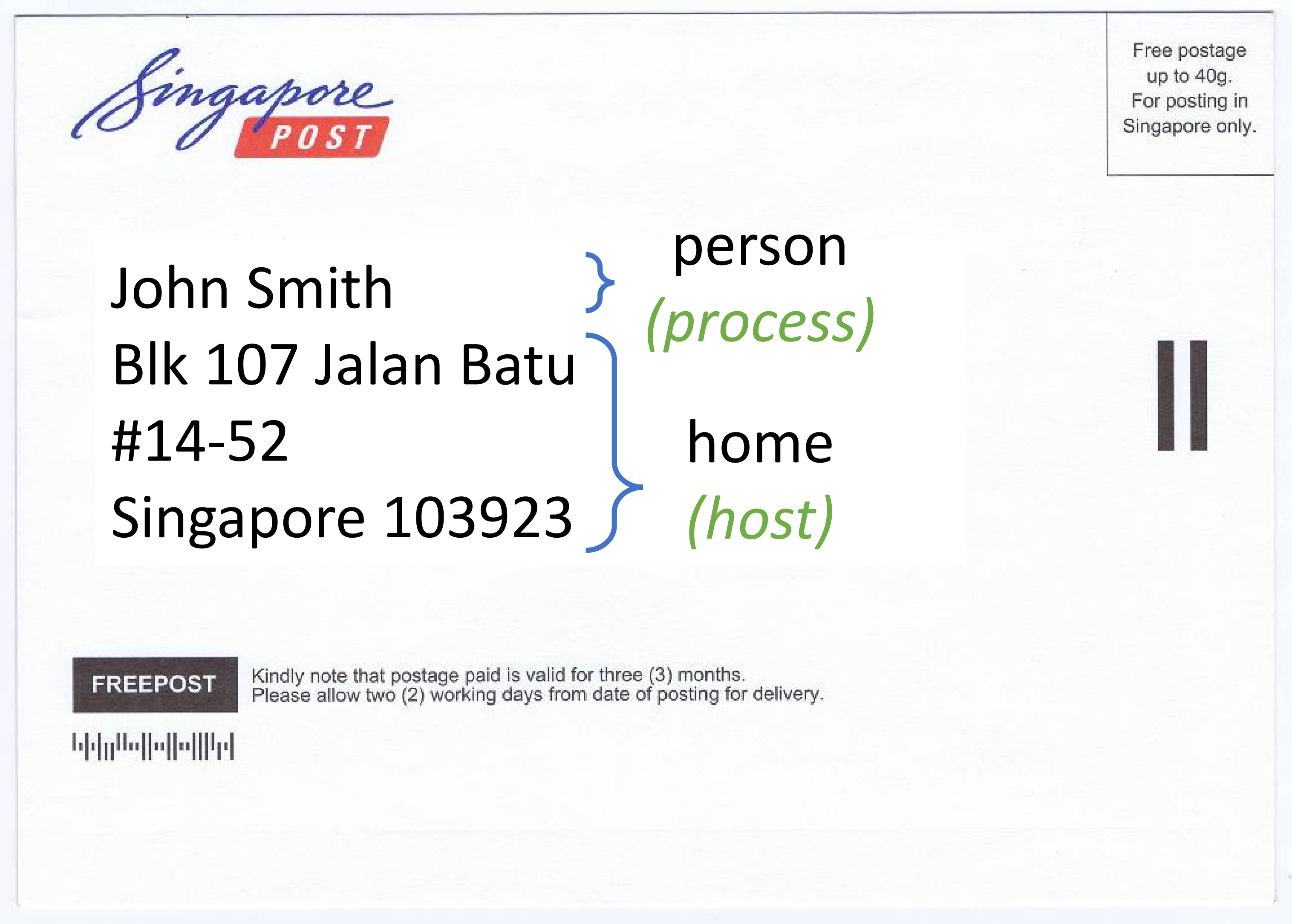
- SingPost? DHL? FedEX?

How will you interact with the delivery service?

- Envelope? Parcel?

How will the delivery service know where to deliver?

- Address of destination
- But address only gets to the home
- How about the recipient?


Singapore POST
Free postage up to 40g. For posting in Singapore only.
John Smith
Blk 107 Jalan Batu
#14-52
Singapore 103923
person
(process)
home
(host)
FREEPOST
Kindly note that postage paid is valid for three (3) months.
Please allow two (2) working days from date of posting for delivery.

Identifying network process

A host can have several processes (apps)

- How to identify which process/app to send the data?

IP Address

- identifies the host
- 32-bit integers written with 4 numbers, e.g. 192.168.0.1

port number

- identifies the process
- 16-bit integers (1 to 65535)
- 1 to 1023 are reserved

IANA assigns the port numbers

http://www.ietf.org/assignments/port-numbers

What transport service does an application need?

Data integrity

- 100% reliable or loss-tolerable?

Timing

- time critical or not?

Throughput

- minimum throughput or elastic?

Security (lecture 8)

Internet Transport Protocols

TCP & UDP

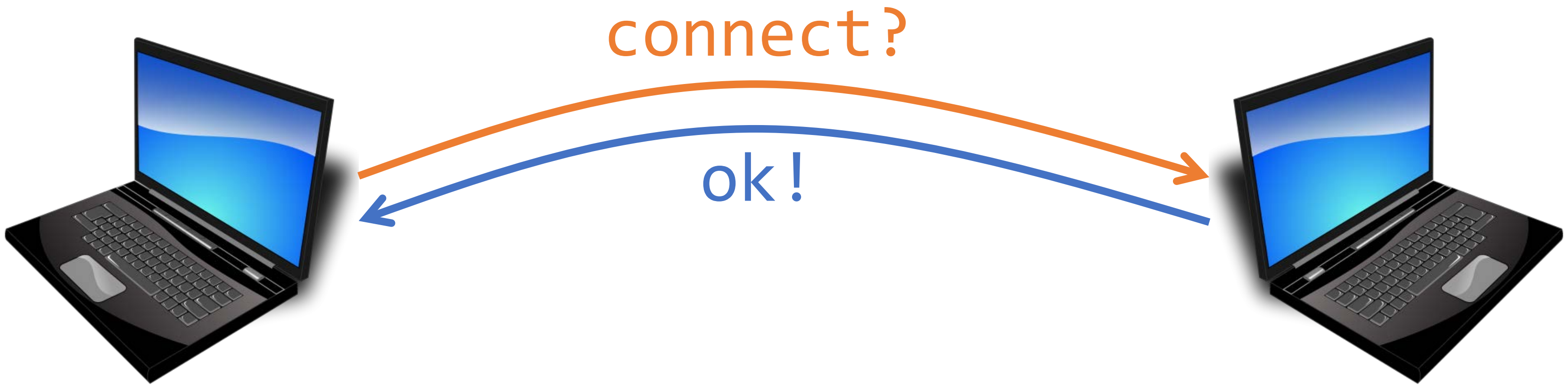
(Lecture 5)

TCP is

- connection oriented
- flow controlled
- congestion controlled
- reliable

TCP is connection oriented

Time is needed to setup a connection



P.S. this is a gross simplification. More details in Lecture 5.

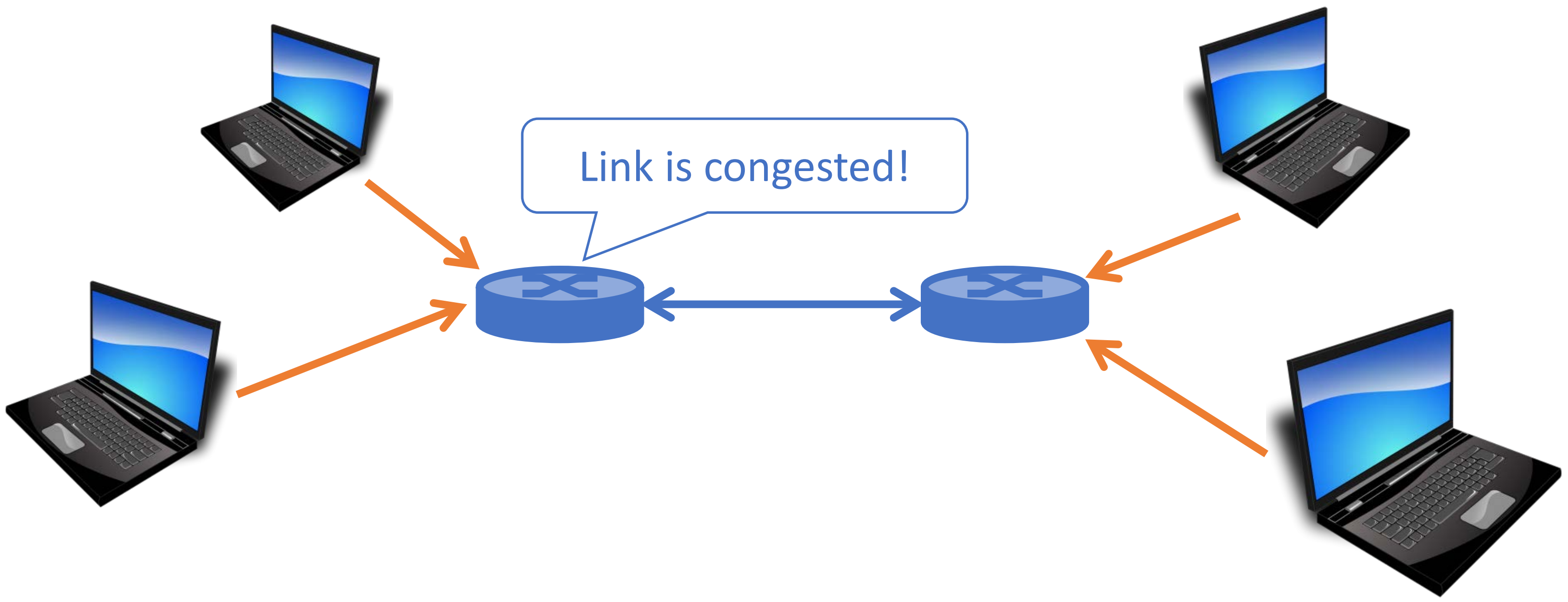
TCP has flow control

Prevents sender from flooding receiver

Again more details in Lecture 5

TCP has congestion control

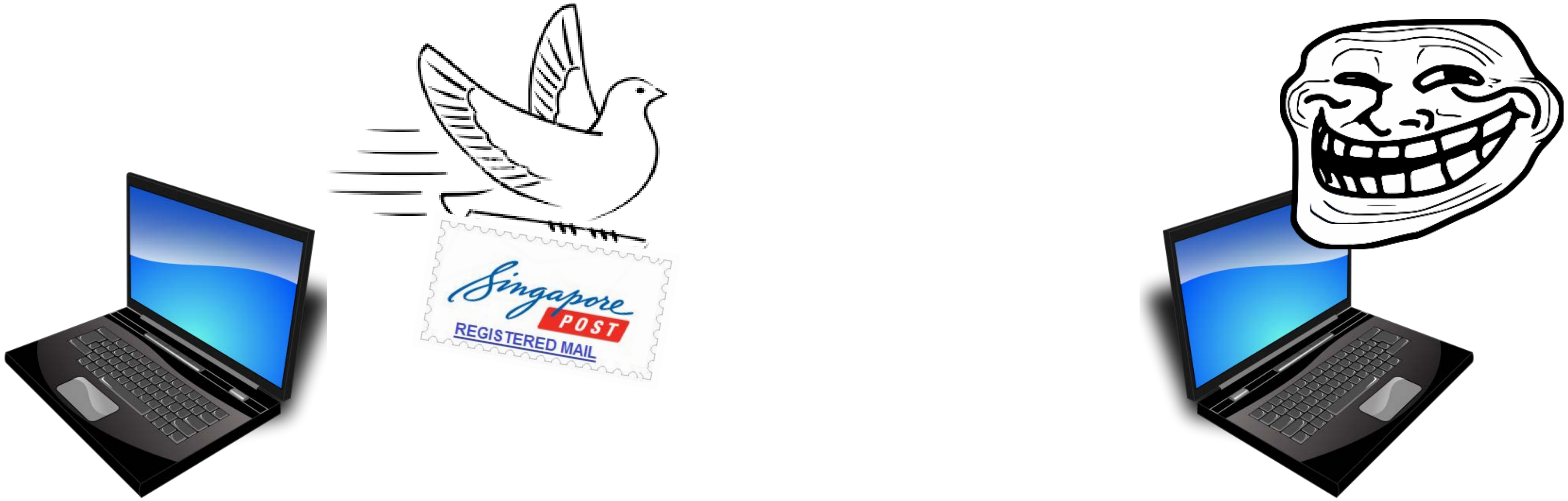
Mitigates congestion



Not covered in CS2105.
Chapters 3.6 & 3.7 of Kurose and Ross.
Covered in CS3103.

TCP is reliable

Guaranteed delivery of data



However, no guarantees on throughput or delay

UDP is

None of the above

Internet Transport Protocols

TCP service:

- reliable transport between sending and receiving process
- flow control: sender won't overwhelm receiver
- congestion control: throttle sender when network is overloaded
- does not provide: timing, minimum throughput guarantee, security

UDP service:

- unreliable data transfer between sending and receiving process
- does not provide: reliability, flow control, congestion control, timing, throughput guarantee or security

When writing network applications, we must ask:

what architecture?

what type of service?

how to exchange messages?

Lecture 2: Roadmap

2.1 Principles of Network Applications

2.2 Web and HTTP

2.5 DNS

2.7 Socket programming with TCP

2.8 Socket programming with UDP


World Wide Web

HTTP and the Web

Hyper-text Transfer Protocol

What is Hyper-text?

Hyper-text Mark-up Language

(HTML)

What is a web page?

An HTML file with several other objects

Web objects are addressable by a URL

`http://www.comp.nus.edu.sg/~cs2105/img/doge.jpg`

The Web: Some Jargon

A Web page typically consists of:

- base HTML file, and
- several referenced objects.

An object can be HTML file, JPEG image, Java applet, audio file, …

Each object is addressable by a URL, e.g.,

$$\underbrace{\texttt{www.comp.nus.edu.sg}}_{\text{host name}}\underbrace{\texttt{/\textasciitilde cs2105/img/doge.jpg}}_{\text{path name}}$$

HTTP Overview

HyperText Transfer Protocol

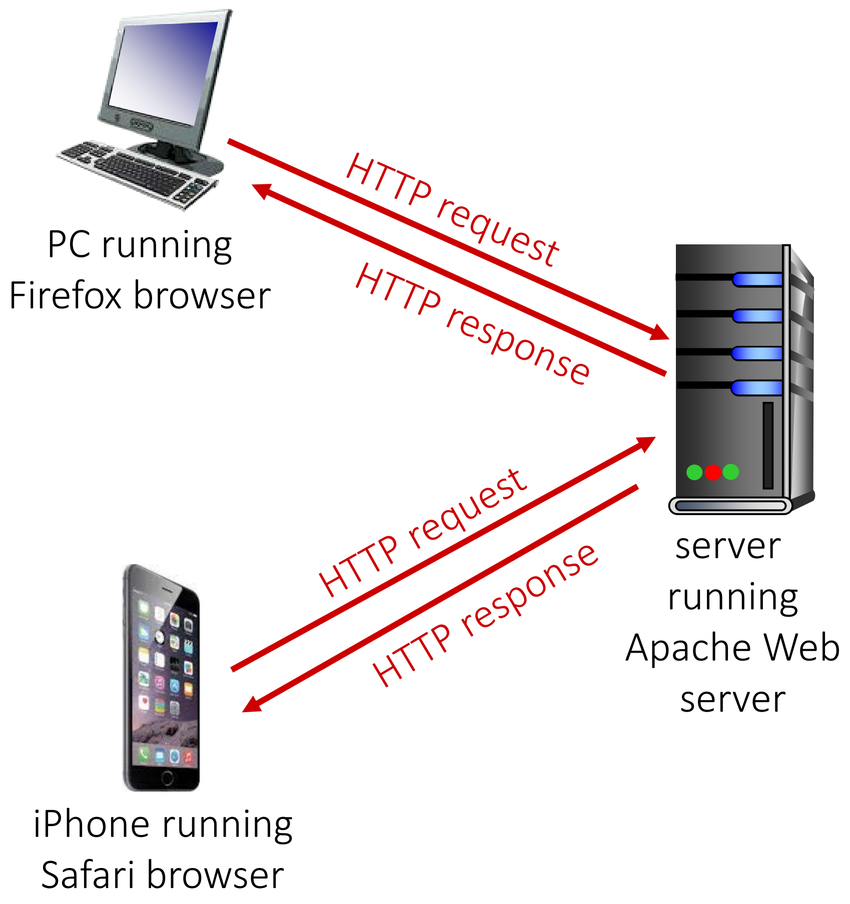
Web’s application layer protocol

Client/server model

- client: usually is browser that requests, receives and displays Web objects
- server: Web server sends objects in response to requests

http 1.0: RFC 1945

http 1.1: RFC 2616



HTTP/1.0

HTTP/1.1

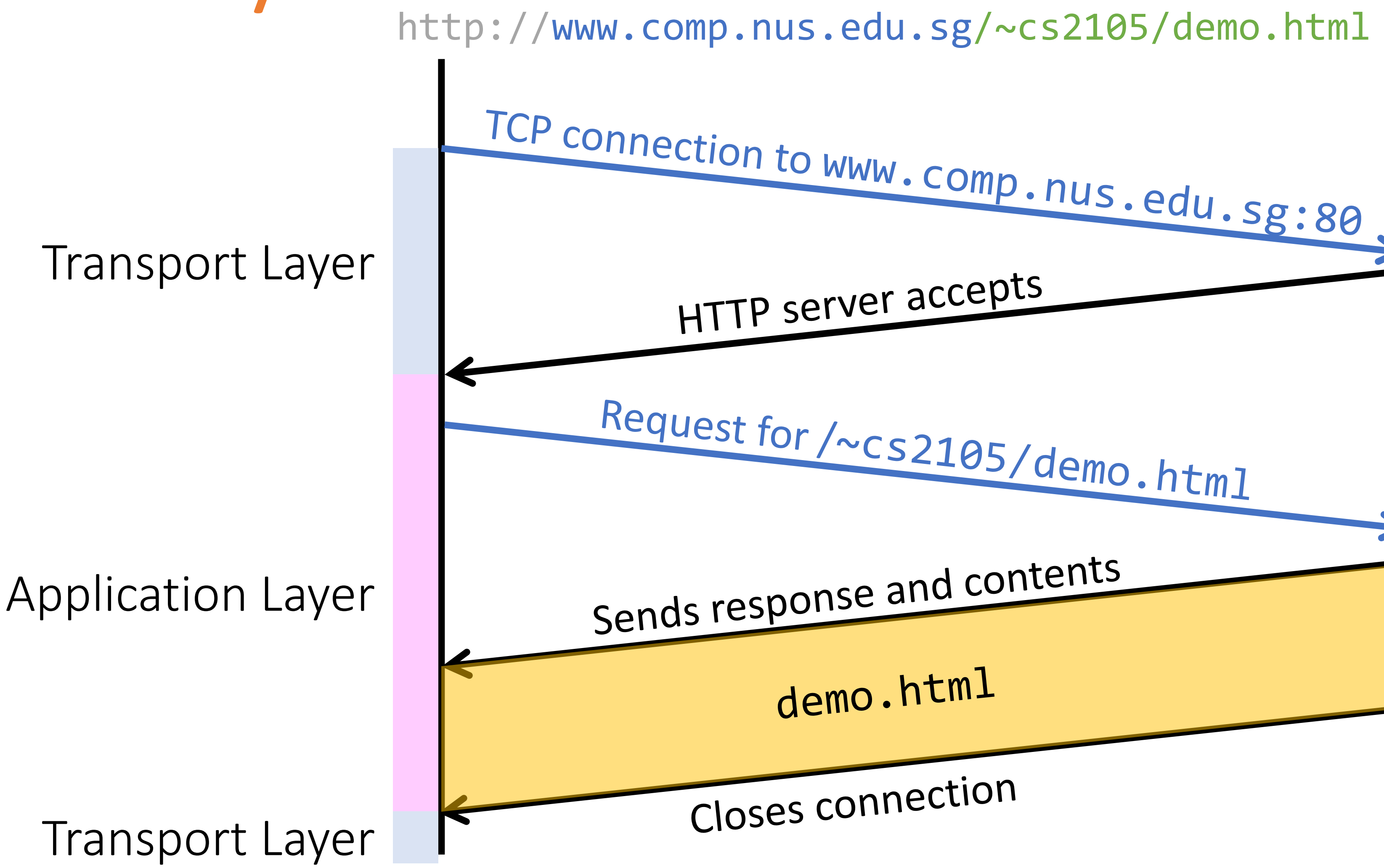
HTTP/2

HTTP Over TCP

HTTP uses TCP as transport service

- Client initiates TCP connection to server.
- Server accepts TCP connection request from client.
- HTTP messages are exchanged between browser (HTTP client) and Web server (HTTP server) over TCP connection.
- TCP connection closed.

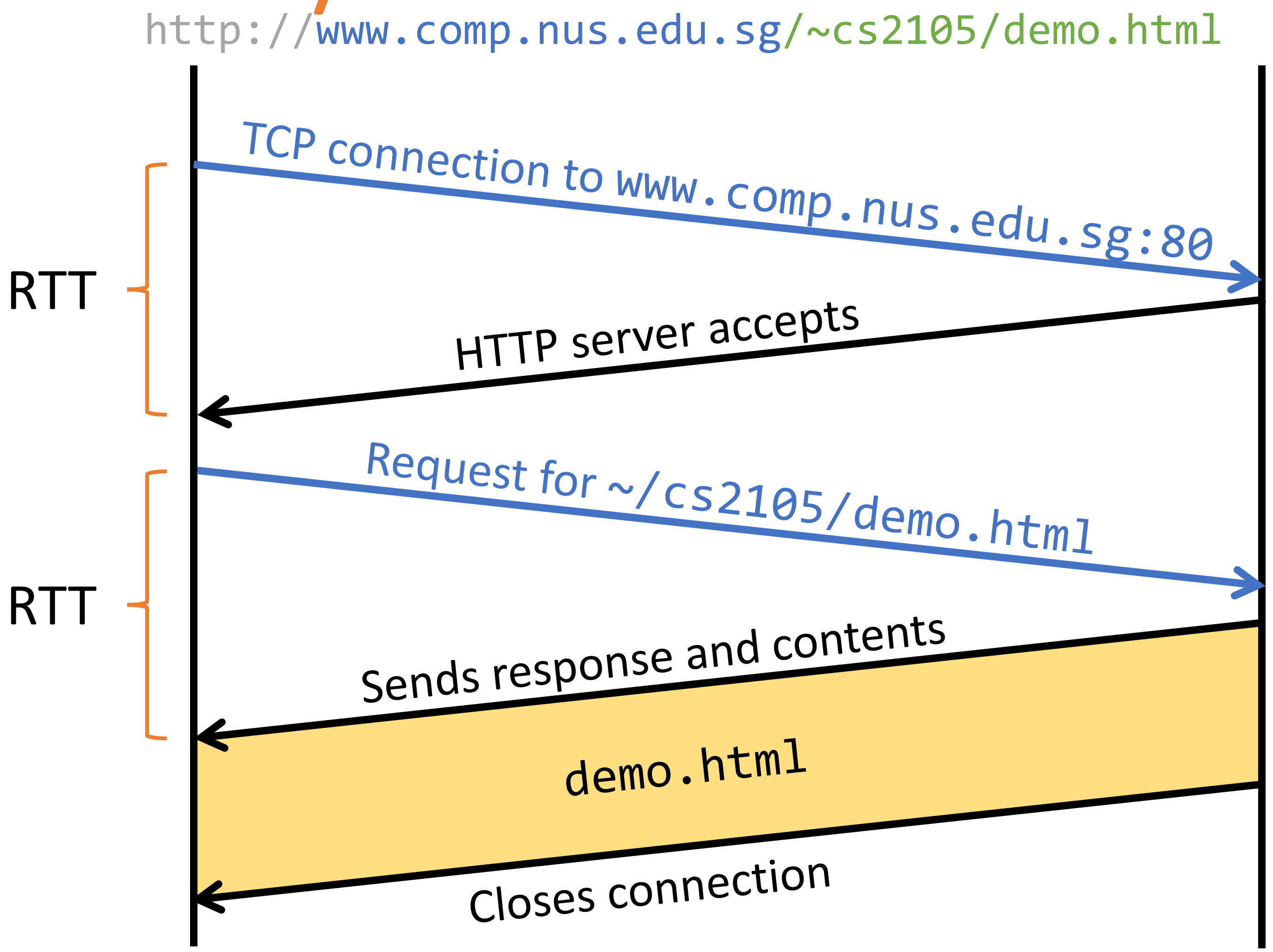
HTTP/1.0 Overview



whole process repeats for each object in the html file

Round-Trip Time
(RTT)

HTTP/1.0 Overview



RTT: time for a packet to travel from client to server and go back

HTTP response time:

- one RTT to establish TCP connection
- one RTT for HTTP request and the first few bytes of HTTP response to return
- file transmission time
- non-persistent HTTP response time = 2 * RTT+ file transmission time

whole process repeats for each object in the html file

HTTP/1.0

vs.

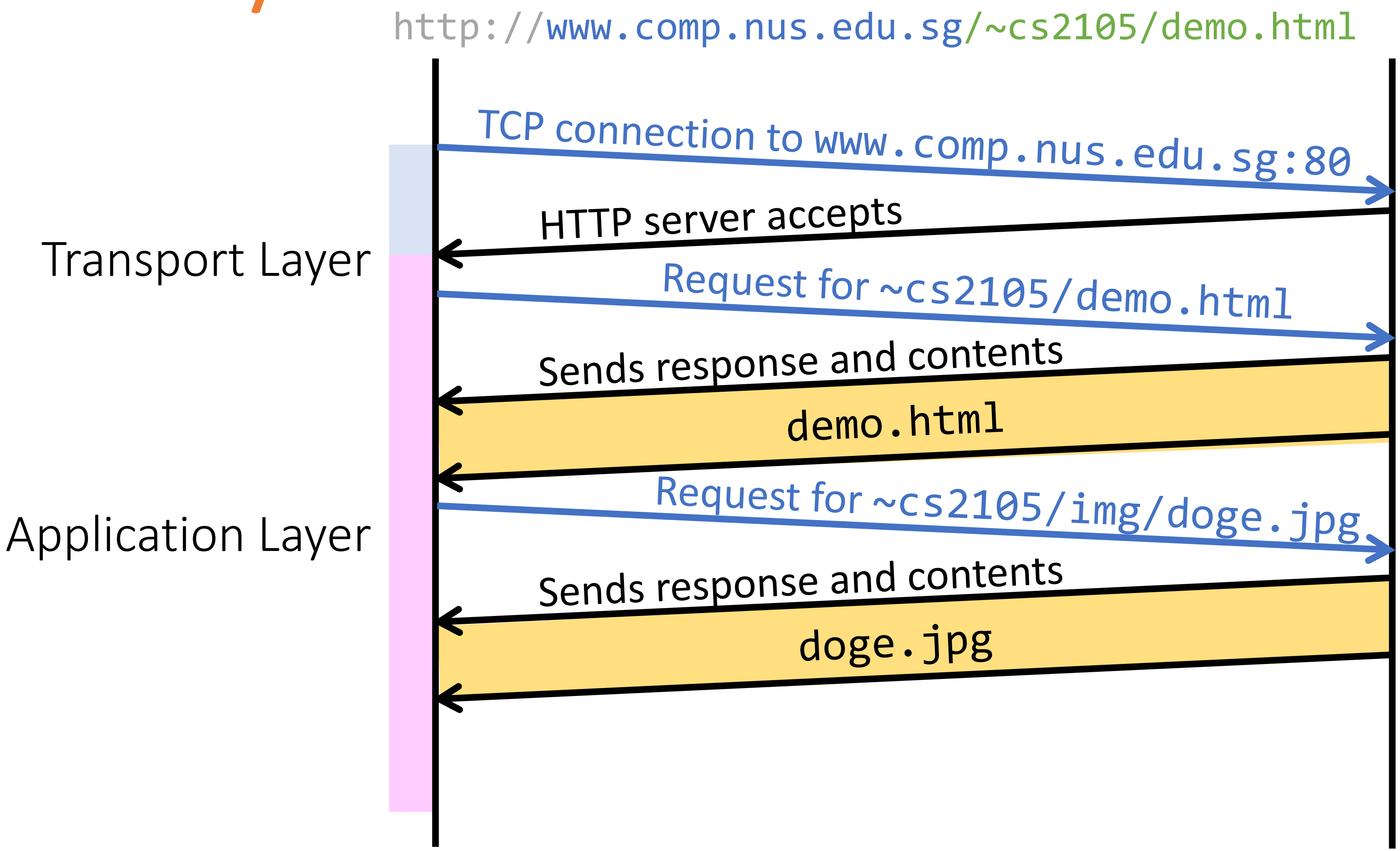
HTTP/1.1

Non-persistent

vs.

Persistent

HTTP/1.1 Overview



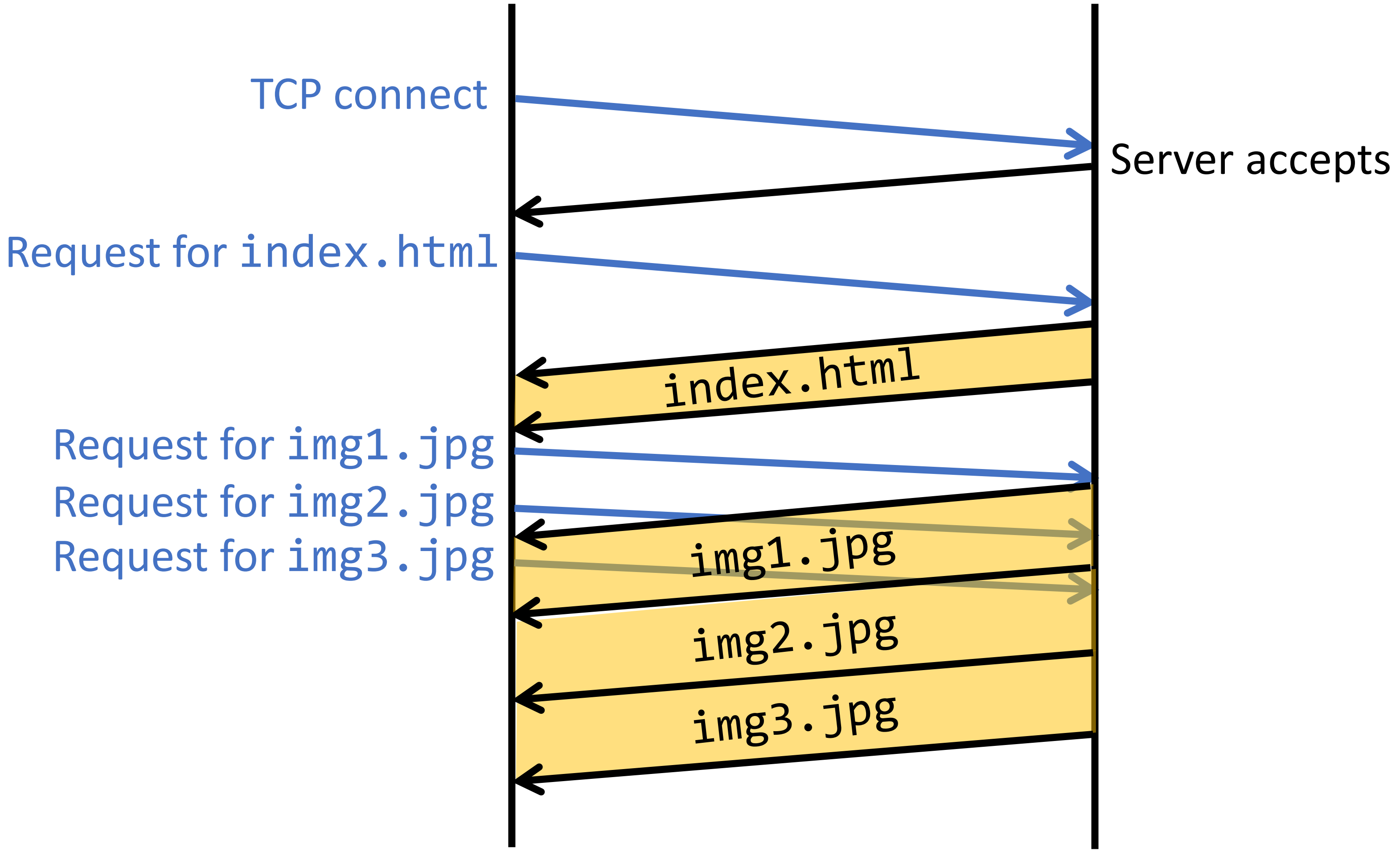
whole process repeats for each object in the html file

Pipeline

vs.

Sequential

HTTP/1.1 Pipelining



New request made even before receiving response of old requests

Persistent HTTP

non-persistent HTTP issues:

- requires 2 RTTs per object
- OS overhead for each TCP connection
- browsers often open parallel TCP connections to fetch referenced objects

persistent HTTP:

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over the same TCP connection
- client sends requests as soon as it encounters a referenced object (persistent with pipelining)
- as little as one RTT for all the referenced objects

Pipeline

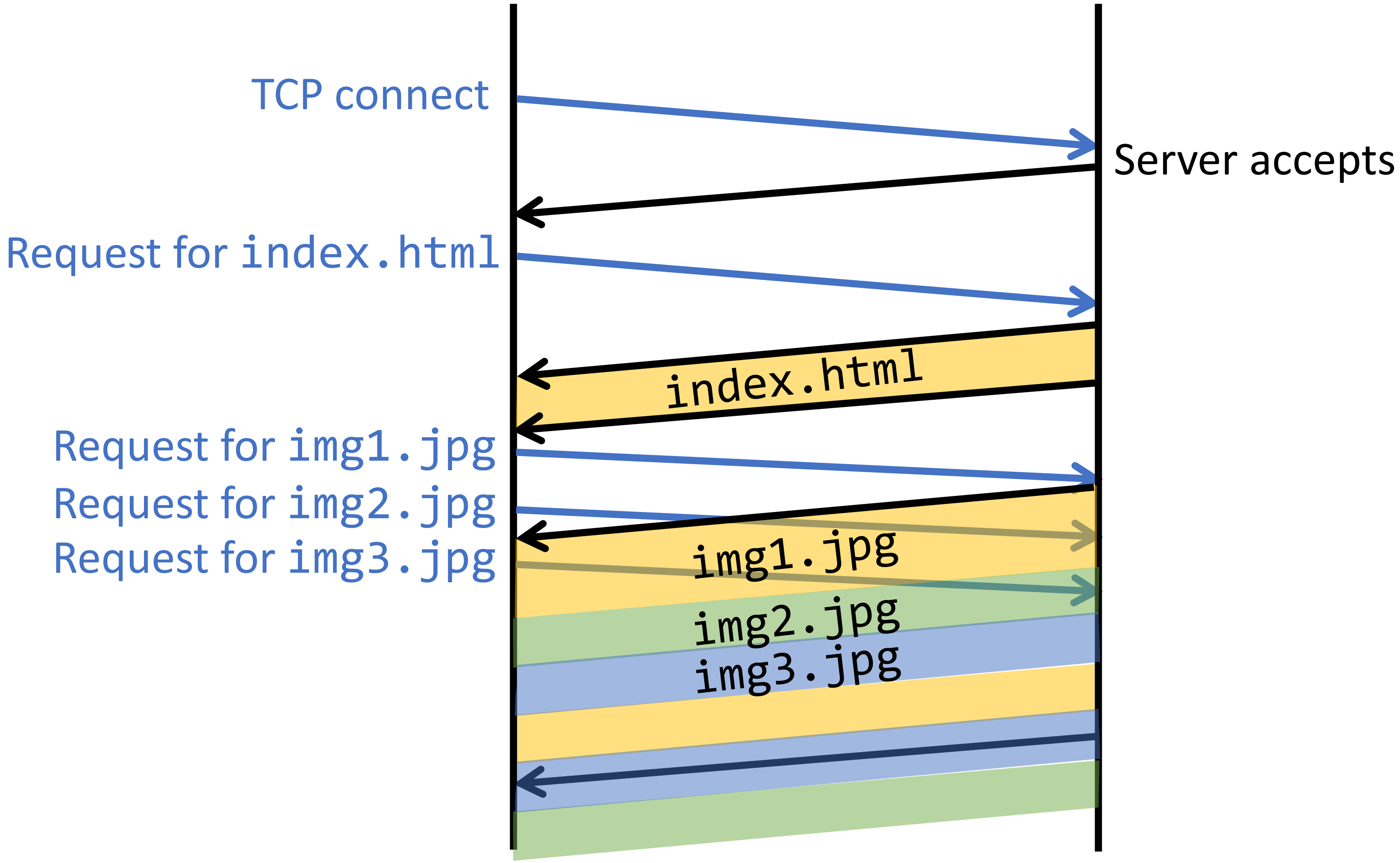
vs.

Sequential

vs.

Multiplexing

HTTP/2 Multiplexing



Response can come back in any order, even partially.

HTTP Request

Request Type: GET method

```
GET /~cs2105/demo.html HTTP/1.1 \r\n
Host: www.comp.nus.edu.sg \r\n
User-Agent: Mozilla/5.0 \r\n
Connection: close \r\n
\r\n
```

All lines ends with this

All browsers today call themselves Mozilla
http://webaim.org/blog/user-agent-string-history/

Blank line marks end of headers

For a full list of HTTP Headers, see
www.w3.org/Protocols/rfc2616/rfc2616-sec14.htm

HTTP Response

Status line: Protocol and response code

```
HTTP/1.1 200 OK
Date: Wed, 01 Jul 2015 08:47:52 GMT
Server: Apache/2.4.6 (Unix) OpenSSL/1.0.1m
Accept-Ranges: bytes
Connection: Keep-Alive
Content-Length: 73
Content-Type: text/html
Keep-Alive: timeout=5, max=100

<!DOCTYPE html>
<html lang="en">
…
```

Blank line marks end of header

Data, e.g. requested HTML file

telnet

curl

Plain Text (ASCII)

vs

Binary

HTTP Status Code

200 Ok

- Request successful, requested object follows

301 Moved Permanently

- New location to follow

304 Not Modified

- Object has not changed since specified date/time

403 Forbidden

- Server declines to show the webpage

404 Not Found

- Requested object not found

500 Internet Server Error

- Unspecified error

Stateless

vs.

Stateful

HTTP was designed to be stateless.

Cookies are used to maintain state.

Cookies

HTTP is designed to be “stateless”.

- Server maintains no information about past client requests.

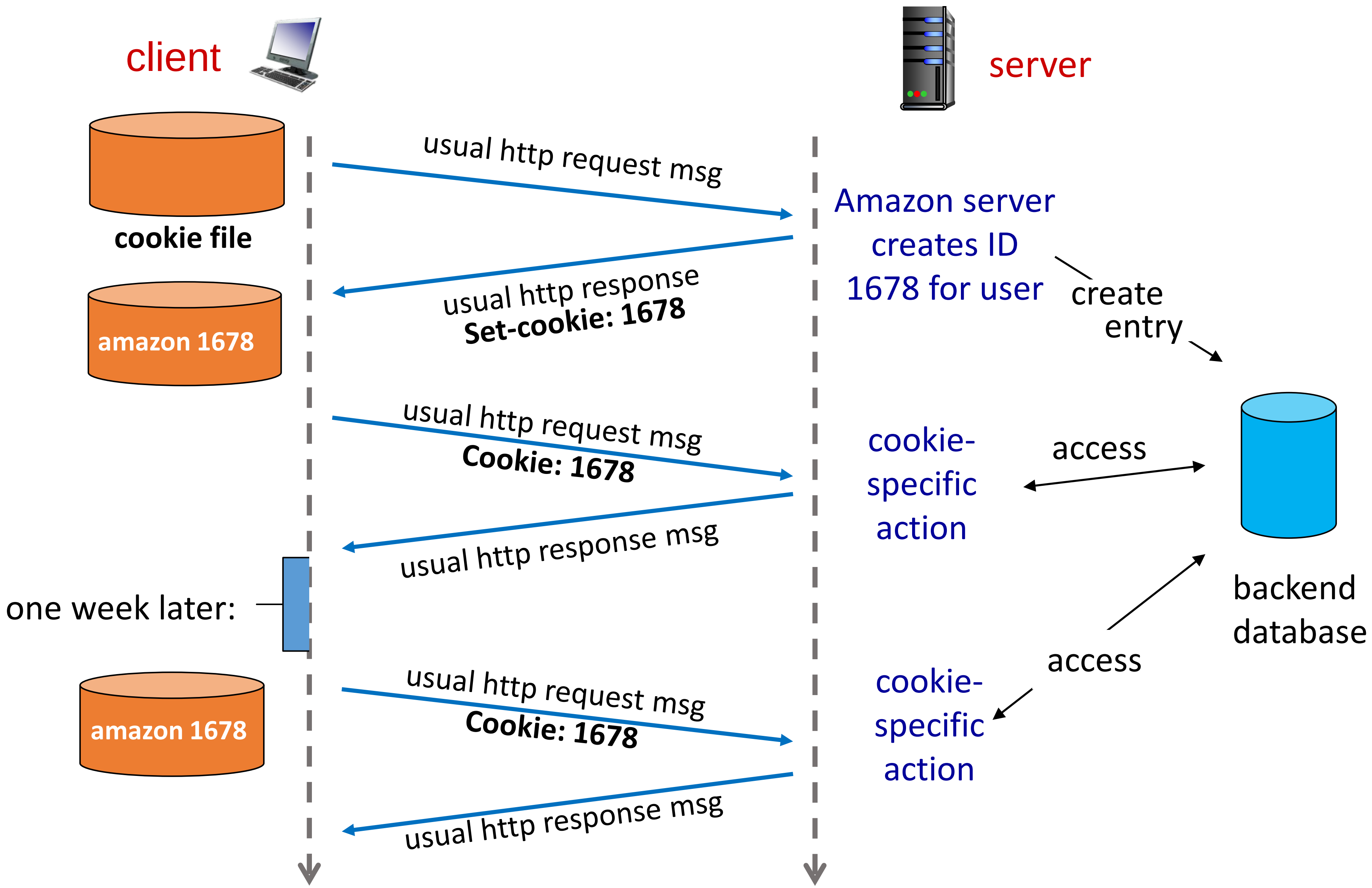
Sometimes it’s good to maintain states (history) at server/client over multiple transactions.

- E.g. shopping carts, login account

Cookie: http messages carry “state”

1. cookie header field of HTTP request / response messages
2. cookie file kept on user’s host, managed by user’s browser
3. back-end database at Web site

Keeping User State with Cookie



Caching web resources

No need to keep downloading resources if nothing has changed

- Images
- Javascripts
- Cascading Style Sheets

How to tell if resource has changed?

Conditional GET

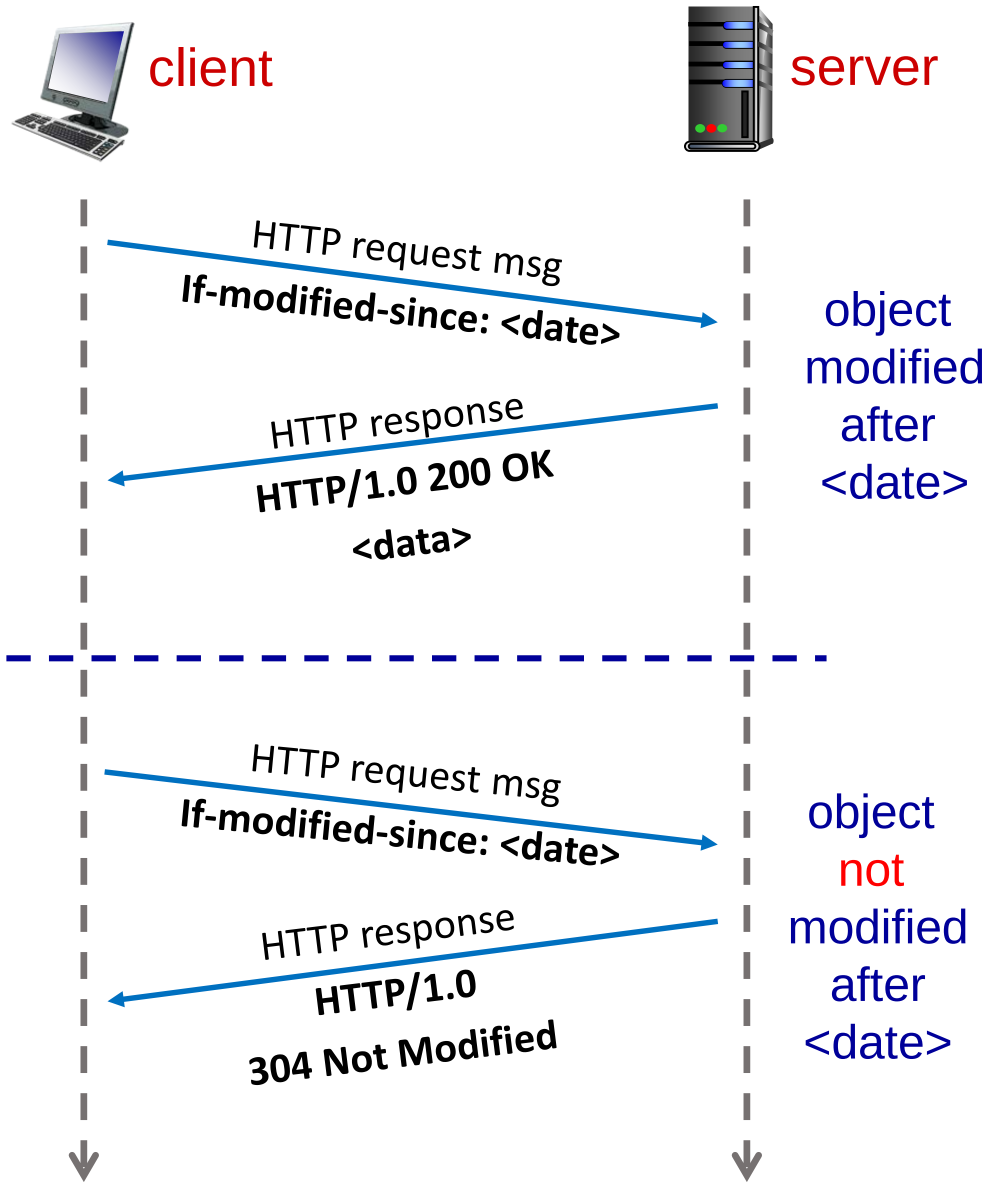
Goal: don't send object if (client) cache has up-to-date cached version

cache: specify date of cached copy in HTTP request

```
If-modified-since:
   <date>
```

server: response contains no object if cached copy is up-to-date:

```
HTTP/1.0 304 Not
   Modified
```



<break>

Lecture 2: Roadmap

2.1 Principles of Network Applications

2.2 Web and HTTP

2.5 DNS

2.7 Socket programming with TCP

2.8 Socket programming with UDP

Earlier, we said that hosts are addressed by their IP address.

How then does a URL identify a host?

DNS

Domain Name Service

DNS translates between host name and IP address

Domain Name System

Two ways to identify a host:

- Hostname, e.g., www.comp.nus.edu.sg
- IP address, e.g., 137.132.80.57

DNS (Domain Name System) translates between the two.

- A client must carry out a DNS query to determine the IP address corresponding to the server name (e.g., www.comp.nus.edu.sg) prior to the connection.

DNS Resource Record

Mapping between host names and IP addresses (and others) are stored as Resouce Records (RR)

`RR Format: <name, value, type, TTL>`

Type	Name	Value
A (adress)	Hostname	IP Address
NS (name server)	Domain, e.g nus.edu.sg	Hostname of authoritative name server for domain
CNAME (canonical name)	Alias for real name, e.g. www.comp.nus.edu.sg	The real name, e.g. www0.comp.nus.edu.sg
MX (mail exchange)	Domain of email address	Name of mail server managing the domain

nslookup

dig

```
suna0 ~>dig comp.nus.edu.sg any

; <<>> DiG 9.6-ESV-R11-S10 <<>> comp.nus.edu.sg any
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30715
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 12, AUTHORITY: 0,
   ADDITIONAL: 9

;; QUESTION SECTION:
;comp.nus.edu.sg.               IN      ANY

;; ANSWER SECTION:
comp.nus.edu.sg.        86400   IN      SOA
   ns1.comp.nus.edu.sg. root.ns1.comp.nus.edu.sg. 2013002696
   3600 1800 604800 86400
comp.nus.edu.sg.        86400   IN      NS
   ns2.nus.edu.sg.
comp.nus.edu.sg.        86400   IN      NS
   ns2.comp.nus.edu.sg.
comp.nus.edu.sg.        86400   IN      NS
   ns3.comp.nus.edu.sg.
comp.nus.edu.sg.        86400   IN      NS
   ns1.nus.edu.sg.
comp.nus.edu.sg.        86400   IN      NS
   ns1.comp.nus.edu.sg.
comp.nus.edu.sg.        86400   IN      MX      20
   mailgw1.comp.nus.edu.sg.
comp.nus.edu.sg.        86400   IN      MX      20
   postfix1.comp.nus.edu.sg.
comp.nus.edu.sg.        86400   IN      MX      10
   mailgw0.comp.nus.edu.sg.
comp.nus.edu.sg.        86400   IN      MX      10
   postfix0.comp.nus.edu.sg.
comp.nus.edu.sg.        86400   IN      A       137.132.80.57
comp.nus.edu.sg.        86400   IN      TXT     "google-site-
   verification=U61JZdunoCo6IXf_FANE2hLLgo-iSvBV-25OzKkb5Jo."

;; ADDITIONAL SECTION:
ns1.nus.edu.sg.         4734    IN      A       137.132.123.4
ns1.comp.nus.edu.sg.    86400   IN      A       137.132.90.2
ns2.nus.edu.sg.         3465    IN      A       137.132.5.2
ns2.comp.nus.edu.sg.    86400   IN      A       137.132.85.2
ns3.comp.nus.edu.sg.    86400   IN      A       137.132.87.2
mailgw0.comp.nus.edu.sg. 86400  IN      A       192.168.20.35
postfix0.comp.nus.edu.sg. 86400 IN      A       192.168.21.67
mailgw1.comp.nus.edu.sg. 86400  IN      A       192.168.49.5
postfix1.comp.nus.edu.sg. 86400 IN      A       192.168.21.75

;; Query time: 2 msec
;; SERVER: 137.132.85.2#53(137.132.85.2)
;; WHEN: Mon Aug 20 00:29:11 SGT 2018
;; MSG SIZE  rcvd: 504
```

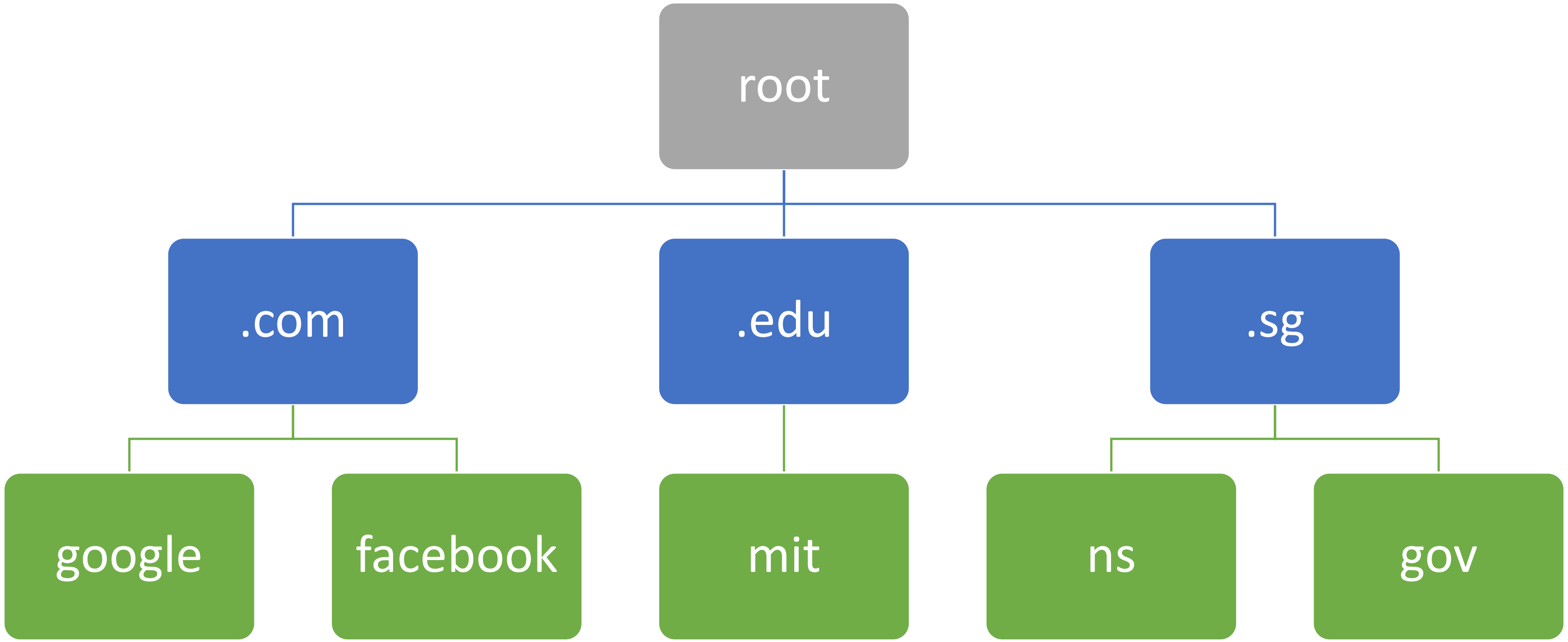
```
suna0 ~>dig www.facebook.com any

; <<>> DiG 9.6-ESV-R11-S10 <<>> www.facebook.com any
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25778
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2,
   ADDITIONAL: 4

;; QUESTION SECTION:
;www.facebook.com.              IN      ANY

;; ANSWER SECTION:
www.facebook.com.       315     IN      CNAME   star-z-
   mini.c10r.facebook.com.

;; AUTHORITY SECTION:
facebook.com.           8173    IN      NS
   a.ns.facebook.com.
facebook.com.           8173    IN      NS
   b.ns.facebook.com.

;; ADDITIONAL SECTION:
a.ns.facebook.com.      51673   IN      A       69.171.239.12
a.ns.facebook.com.      55479   IN      AAAA
   2a03:2880:fffe:c:face:b00c:0:35
b.ns.facebook.com.      51673   IN      A       69.171.255.12
b.ns.facebook.com.      55479   IN      AAAA
   2a03:2880:ffff:c:face:b00c:0:35

;; Query time: 1 msec
;; SERVER: 137.132.85.2#53(137.132.85.2)
;; WHEN: Mon Aug 20 00:38:30 SGT 2018
;; MSG SIZE  rcvd: 188
```

Records are kept by DNS servers.

How do they keep so many records?

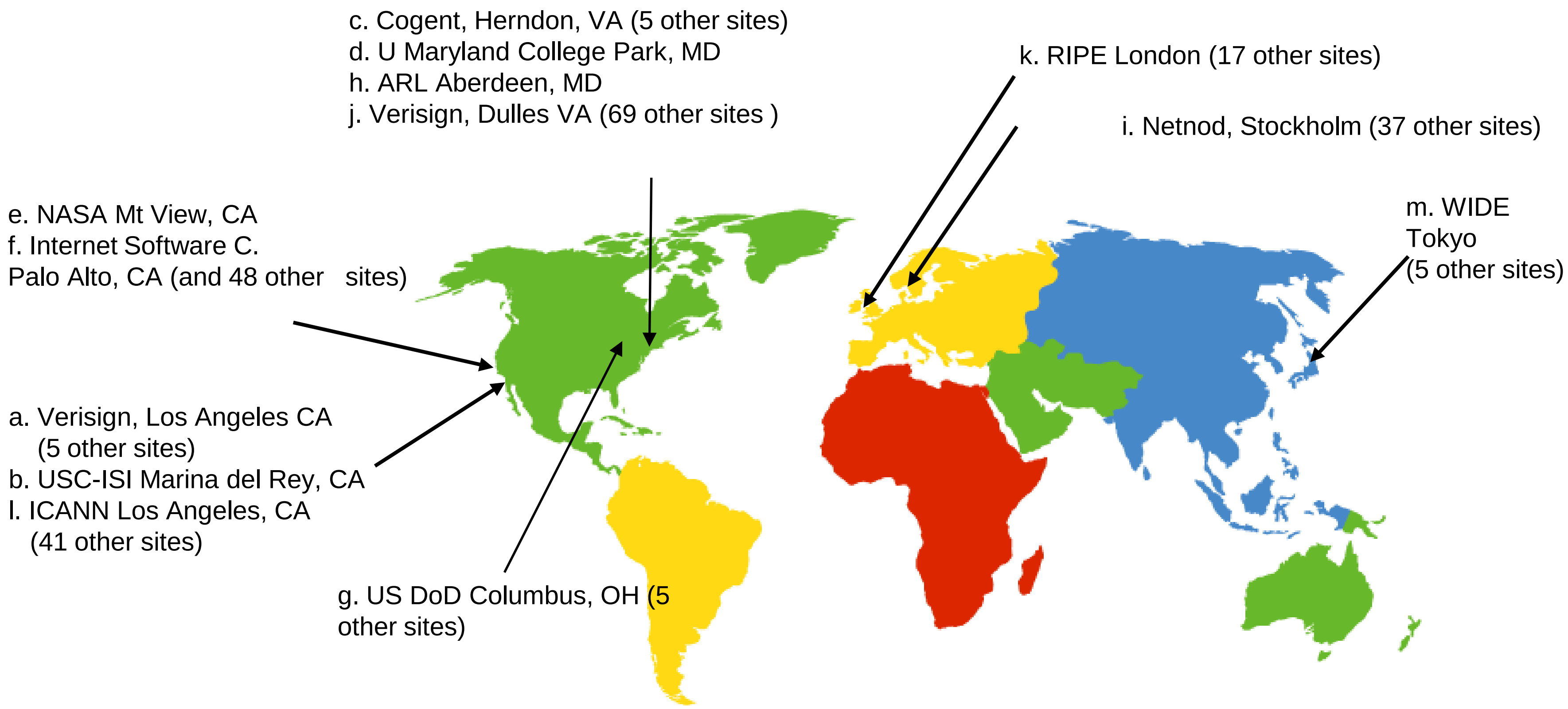
Distributed, Hierarchical Database

DNS stores RR in distributed databases implemented in a hierarchy of many name servers



Root Servers

13 root name servers worldwide



List of root servers

Hostname	IP Addresses	Manager
a.root-servers.net	198.41.0.4, 2001:503:ba3e::2:30	VeriSign, Inc.
b.root-servers.net	192.228.79.201, 2001:500:84::b	University of Southern California (ISI)
c.root-servers.net	192.33.4.12, 2001:500:2::c	Cogent Communications
d.root-servers.net	199.7.91.13, 2001:500:2d::d	University of Maryland
e.root-servers.net	192.203.230.10	NASA (Ames Research Center)
f.root-servers.net	192.5.5.241, 2001:500:2f::f	Internet Systems Consortium, Inc.
g.root-servers.net	192.112.36.4	US Department of Defence (NIC)
h.root-servers.net	128.63.2.53, 2001:500:1::803f:235	US Army (Research Lab)
i.root-servers.net	192.36.148.17, 2001:7fe::53	Netnod
j.root-servers.net	192.58.128.30, 2001:503:c27::2:30	VeriSign, Inc.
k.root-servers.net	193.0.14.129, 2001:7fd::1	RIPE NCC
l.root-servers.net	199.7.83.42, 2001:500:3::42	ICANN
m.root-servers.net	202.12.27.33, 2001:dc3::35	WIDE Project

TLD and Authoritative Servers

Top-level domain (TLD) servers:

- responsible for .com, .org, .net, .edu, …
- and all top-level country domains, e.g., .uk, .sg, .jp

Authoritative servers:

- Organization's own DNS server(s)
- provides authoritative hostname to IP mappings for organization's named hosts (e.g. Web, mail)
- can be maintained by organization or service provider

Local DNS Server

Does not strictly belong to hierarchy

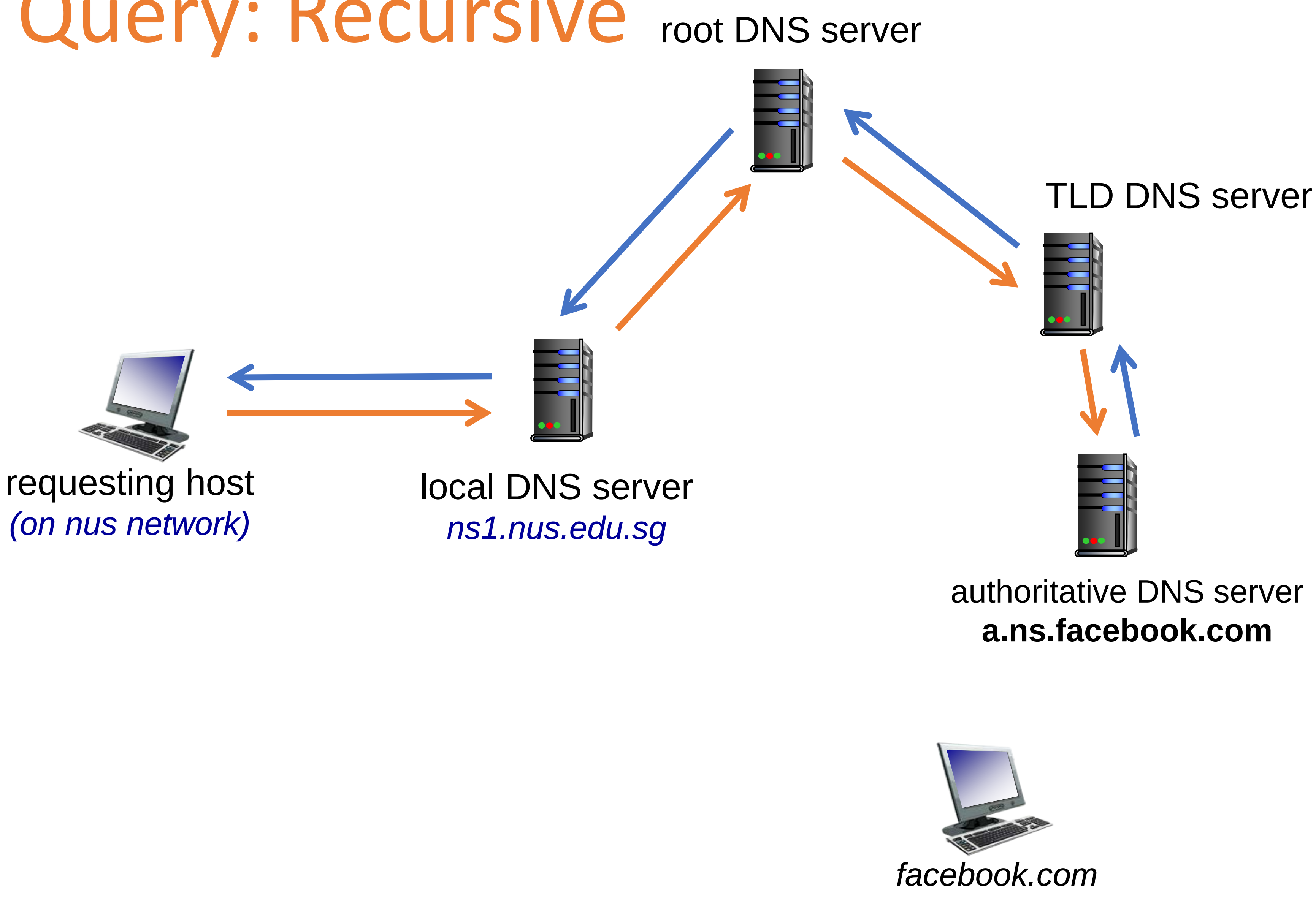
Each ISP (residential ISP, company, university) has one local DNS server.

- also called “default name server”

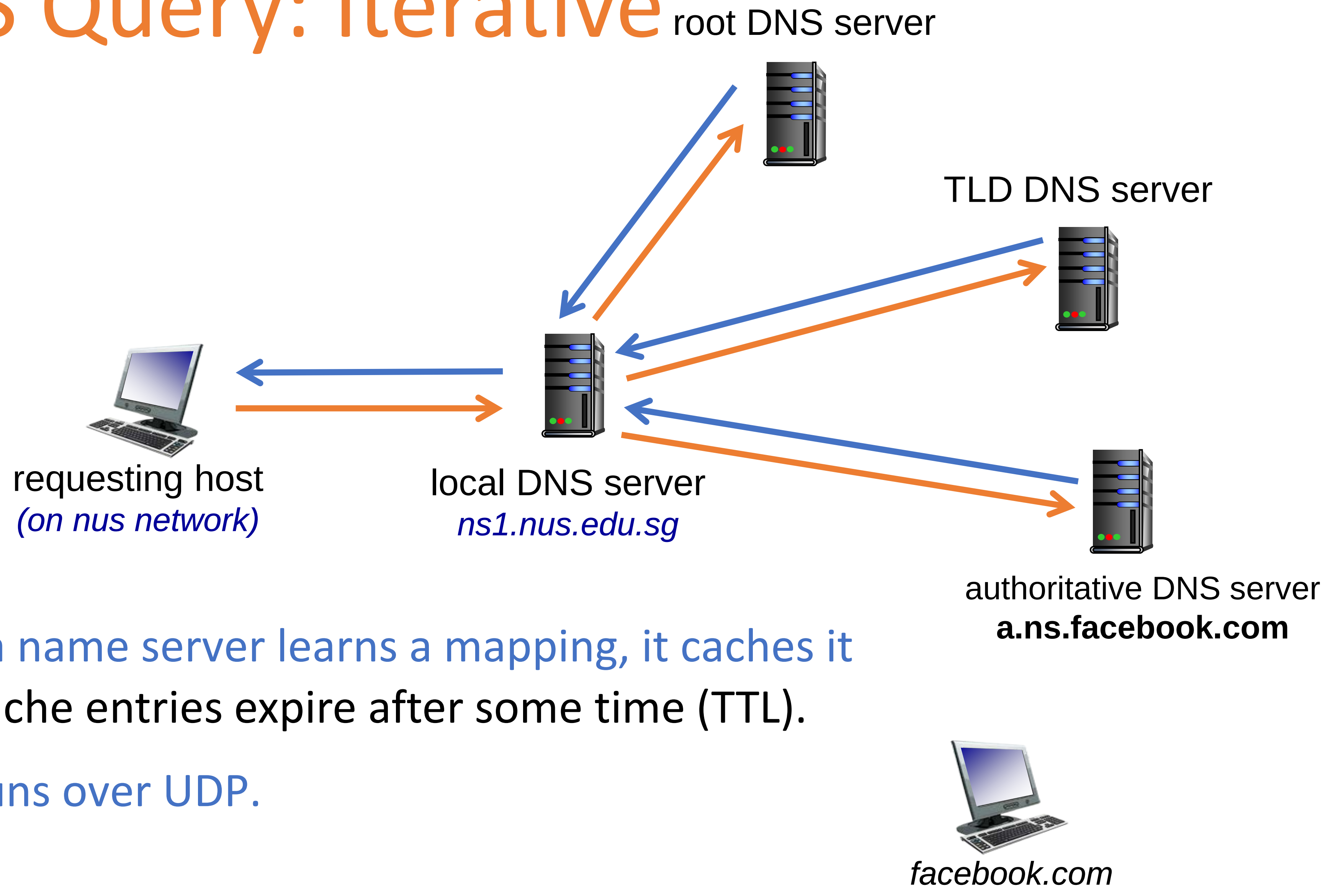
When host makes a DNS query, query is sent to its local DNS server

- Retrieve name-to-address translation from local cache
- Local DNS server acts as proxy and forwards query into hierarchy if answer is not found locally

DNS Query: Recursive



DNS Query: Iterative



Once a name server learns a mapping, it caches it

- cache entries expire after some time (TTL).

DNS runs over UDP.

Lecture 2: Summary

Application architectures

- Client-server
- P2P
- Hybrid

Application service requirements:

- reliability, throughput, delay, security

Specific protocols:

- HTTP
- DNS

Internet transport service model

- TCP : connection-oriented, reliable
- UDP : Connection-less, unreliable