Slides Production: Prof. Yau Kok Lim

Dr. Saad Aslam

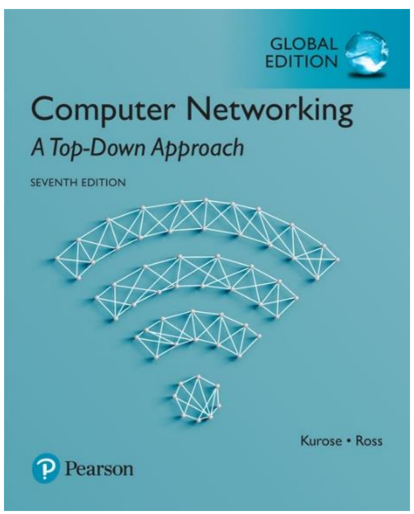
Office: Room AE-327

3rd Floor, NUB

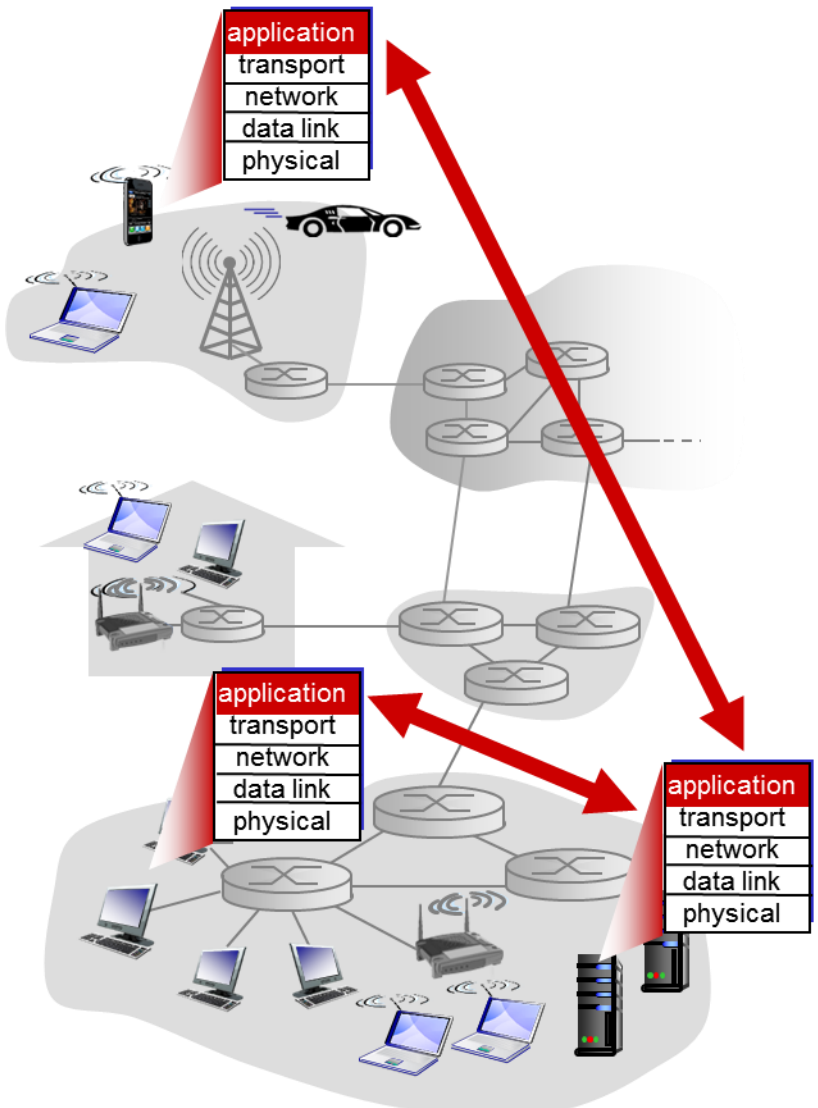
Contact: +603 74918622

(Ext: 7143)

Email: saada@sunway.edu.my

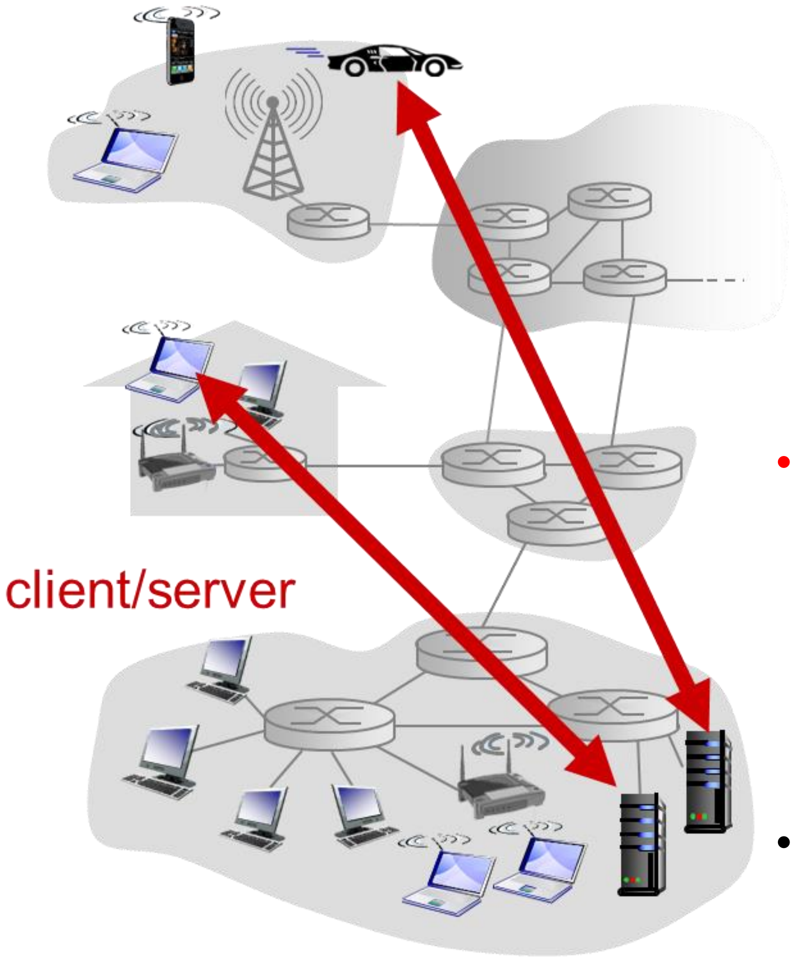


Computer Networking: A Top-down Approach, 7th edition.
Jim Kurose, Keith Ross
Pearson

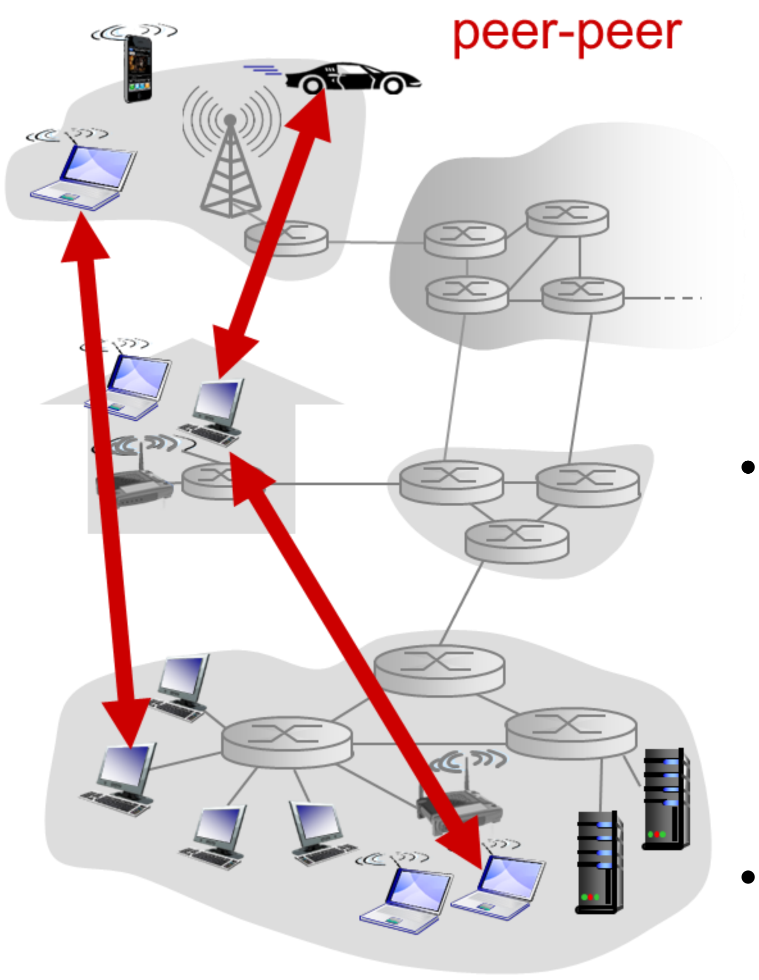


- Examples of Applications
  - Web and HTTP
  - Domain Name System (DNS)
  - Peer-to-peer applications
- Applications run on host
  - Not network core
    - No need to write software for network core
      - Network core do not run user applications
- Application architecture
  - Two types
    - **Client-server architecture**
    - **Peer-to-peer architecture**

2.1 Network Application Architectures: Client-Server Architecture

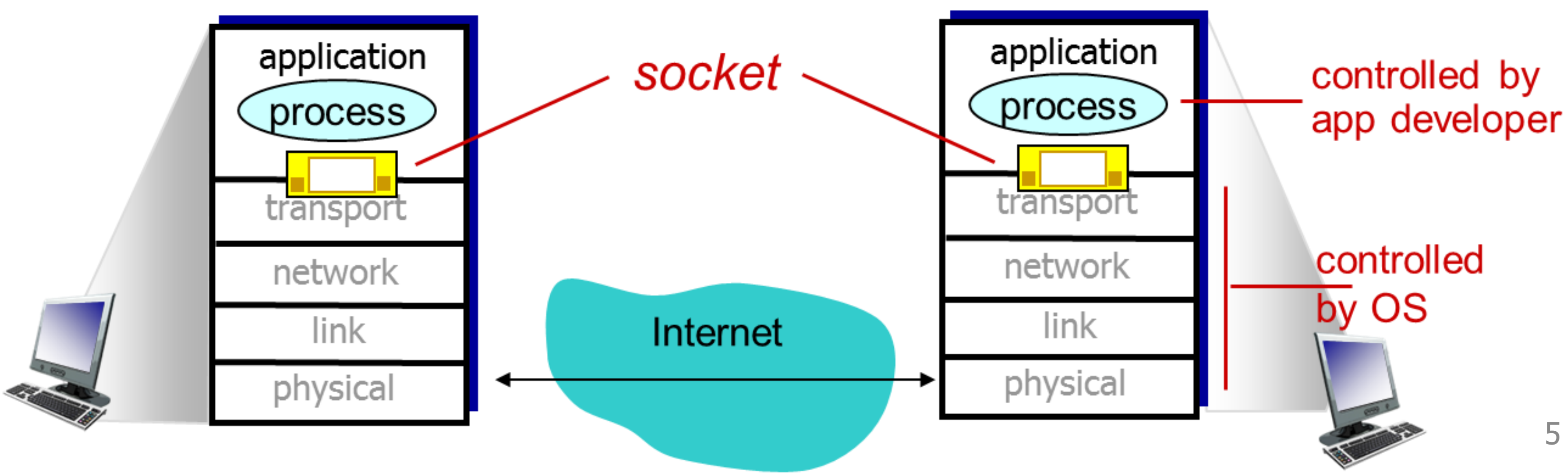


- **Server**
  - Characteristic
    - Always-on host
    - Has permanent IP address
    - Data center
      - Comprised of hundreds to thousands of servers to serve all requests from clients
  - Function
    - Service request from clients
- **Client**
  - Characteristic
    - May have dynamic IP addresses
    - Do not communicate directly with each other
  - Function
    - Send request to server
- Example
  - Web server (server) service request from browser (client)
    - Browsers do not communicate directly with each other



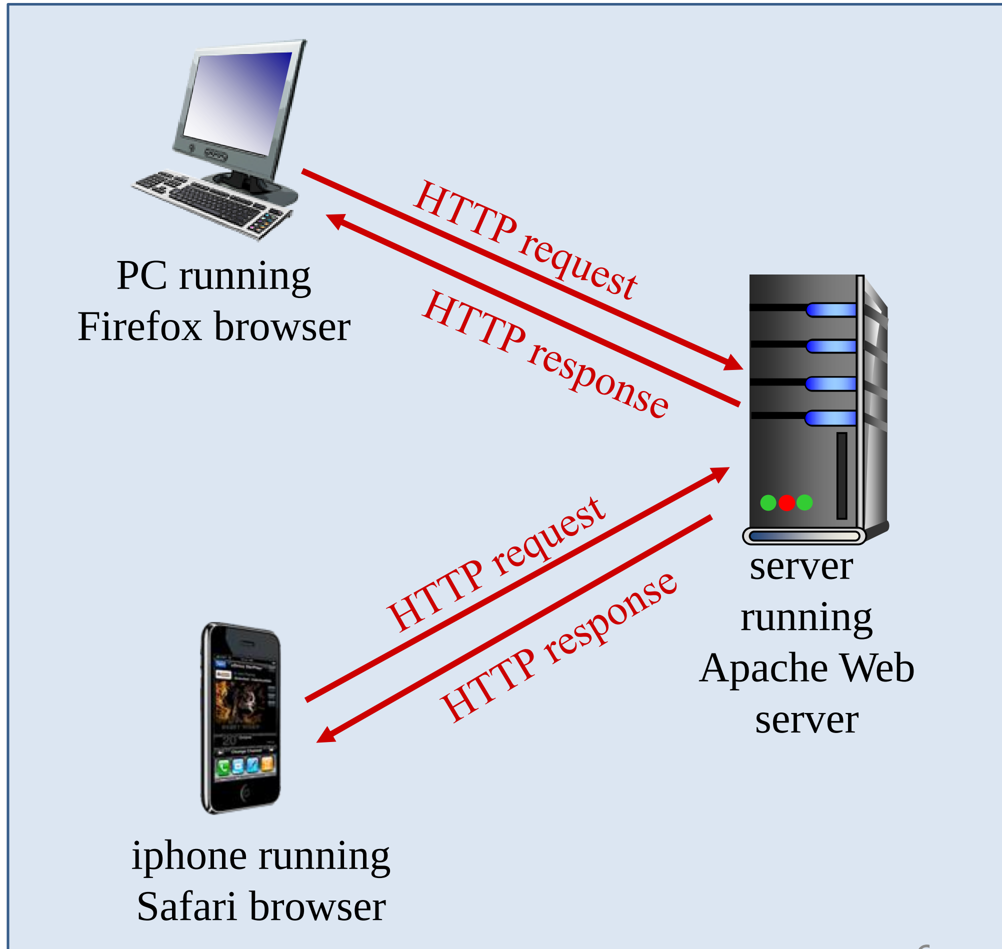
- **Peer**
  - Characteristic
    - Hosts communicate directly with each other
    - No always-on host
  - Function
    - Peer request service from other peer, then provide service to other peer
- Advantage
  - **Self-scalability**
    - New peers provide service to other peer
  - Low cost
    - Expensive server is not necessary
- Example of applications
  - File sharing (BitTorrent)
  - Internet Telephony (Skype)

- **Process**
  - Program running within a host
  - Processes in different hosts communicate by exchanging **messages**
    - Hosts is uniquely identified by IP address (e.g.: 128.119.245.12)
    - Process is uniquely identified by port number (e.g.: 80)
    - Example: Client process in browser send request to server process in web server
- **Socket (Application Programming Interface, API)**
  - Software interface
    - Process sends and receives message to/from its socket
    - Between application layer and transport layer



- **HyperText Transfer Protocol (HTTP)**
  - Application layer protocol for web page
  - Web page consists
    - Base HTML file
    - Object
      - HTML file
      - JPEG image
    - Base HTML file and object are identified by URL (host name and path name)
  - Client-server model
    - client: Web browser requests, receives and displays Web objects
    - server: Web server sends objects in response to requests
  - Two types of messages
    - HTTP request
    - HTTP response

$$\underbrace{\texttt{www.someschool.edu}}_{\text{host name}}\underbrace{\texttt{/someDept/pic.gif}}_{\text{path name}}$$



- **HyperText Transfer Protocol (HTTP)**
  - Use **TCP**
    - Client (browser) initiate TCP connection (creates socket) using port 80 to server
    - Server (web server) accepts TCP connection from client
    - HTTP messages (application-layer protocol messages) exchanged between client and server
    - TCP connection closed
  - Is **stateless**
    - Server maintain no information about past client requests
  - Two types of connections
    - **Non-persistent HTTP connection**
      - Send at most one object over a single TCP connection, then close the connection
      - Require multiple connections to download multiple objects
    - **Persistent HTTP connection**
      - Send multiple objects over a single TCP connection, then close the connection
      - Require a single connection to download multiple objects

2.2 HTTP Connections: Non-Persistent Connection

Suppose user enters a URL that contains text (html file) and references to 10 jpeg images:
www.someSchool.edu/someDepartment/home.index

1a. HTTP client initiate TCP connection at www.someSchool.edu on port 80

1b. HTTP server wait for TCP connection at port 80, accept connection and notify client

2. HTTP client send HTTP **request message** (contain html URL someDepartment/home.index) into TCP connection socket

3. HTTP server receive request message, send **response message** (contain requested html) into TCP connection socket
4. HTTP server close TCP connection

5. HTTP client receive response message (contain html file)
6. Repeat step 1 to 5 for all 10 jpeg objects

time

client

server at www.someSchool.edu

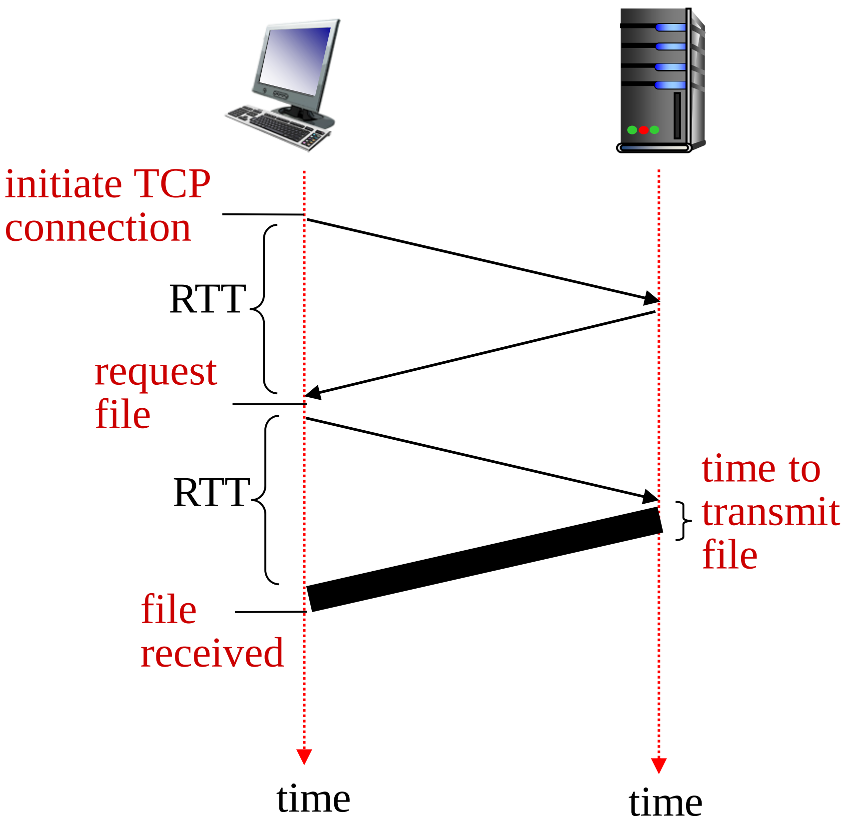
2.2 HTTP Connections: Non-Persistent Connection

- **Round-Trip Time (RTT)**
  - Time for a packet to travel from client to server and then back to client

- **HTTP Response Time**

  = 2RTT + File Transmission Time
  - one RTT to setup TCP connection
  - one RTT for the exchange of HTTP request and HTTP response
  - File transmission time

- Disadvantage
  - Require multiple connections to download multiple objects
    - HTTP response time for each object
    - Solution
      - Client setup parallel TCP connections to receive objects simultaneously



2.2 HTTP Connections: Persistent Connection

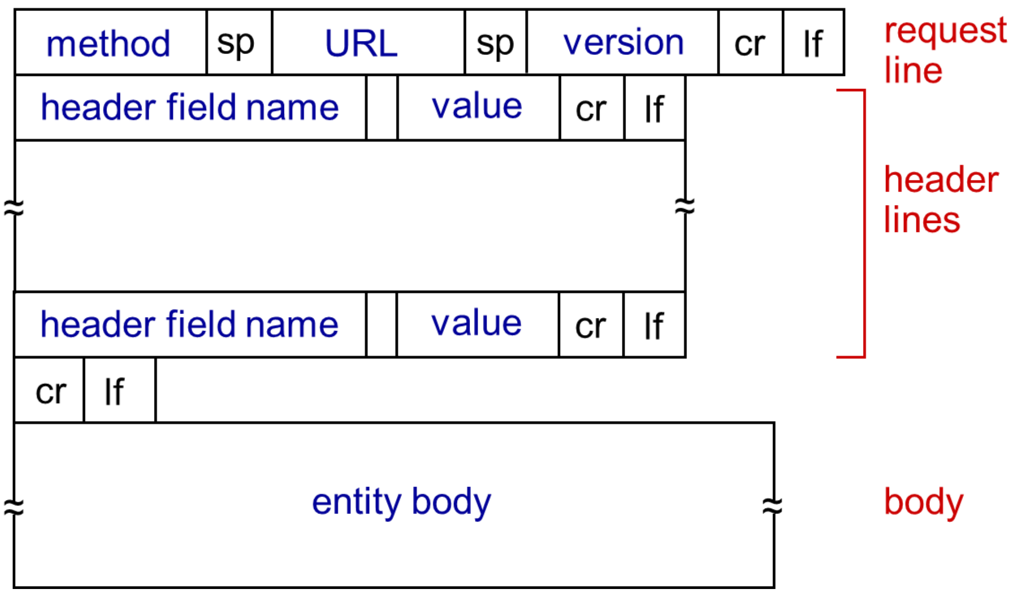
- Persistent HTTP connection
  - Server leaves connection open after sending HTTP response.
    - Subsequent HTTP messages between same client and server sent over the connection
    - The connection close whenever it is not used for a certain time
  - Use **pipelining**
    - Client can send HTTP request and receive HTTP response simultaneously
  - As little as one RTT for all the referenced objects

2.2 HTTP Message Format: HTTP Request

```
GET /index.html HTTP/1.1
Host: www-net.cs.umass.edu
Connection: close
User-Agent: Mozilla/5.0
Accept-Language: fr
```

request line (GET command)

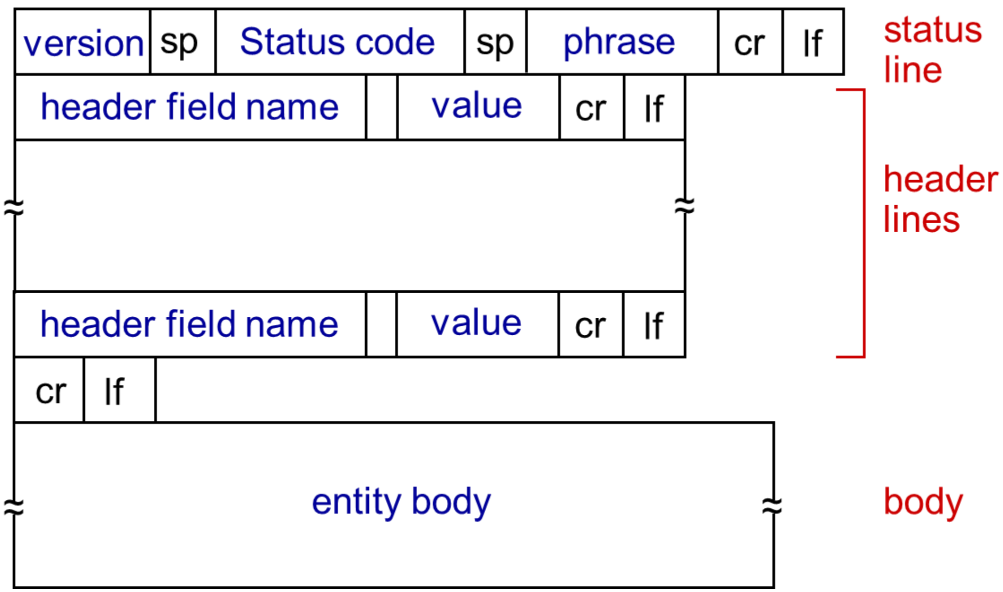
header lines



- Client send HTTP Request to server
- **`GET /index.html HTTP/1.1`**
  - Client request object /index.html using HTTP version 1.1
- **`Host: www-net.cs.umass.edu`**
  - Specify server on which the object reside
- **`Connection: close`**
  - Inform server to close TCP connection after sending the requested object
- **`User-Agent: Mozilla/5.0`**
  - Specify client's browser and version
- Entity body specify contents
  - E.g.: search words in a search engine

2.2 HTTP Message Format: HTTP Response

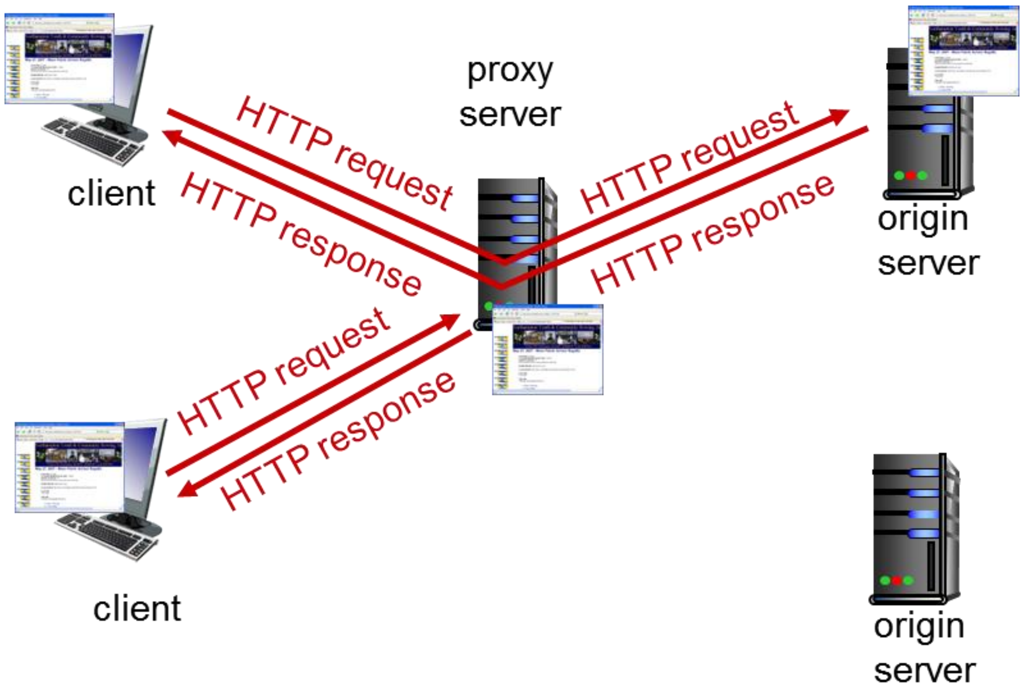
```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 09 Aug 2011 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last Modified: Tue, 09 Aug 2011 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html

(data data data data data …)
```



- Server send HTTP Response to client
- `Date`
  - Time and date when the HTTP response was sent by server
- `Last Modified`
  - Time and date when the object was last modified
- Other types of status code
  - `200 OK`
    - Request is successful. Requested object is specified in body
  - `301 Moved Permanently`
    - Requested object moved. New location is specified in body
  - `400 Bad Request`
    - Indicate error
  - `404 Not Found`
    - Requested object is not found
  - `505 HTTP version not supported`

2.2 Web Caching

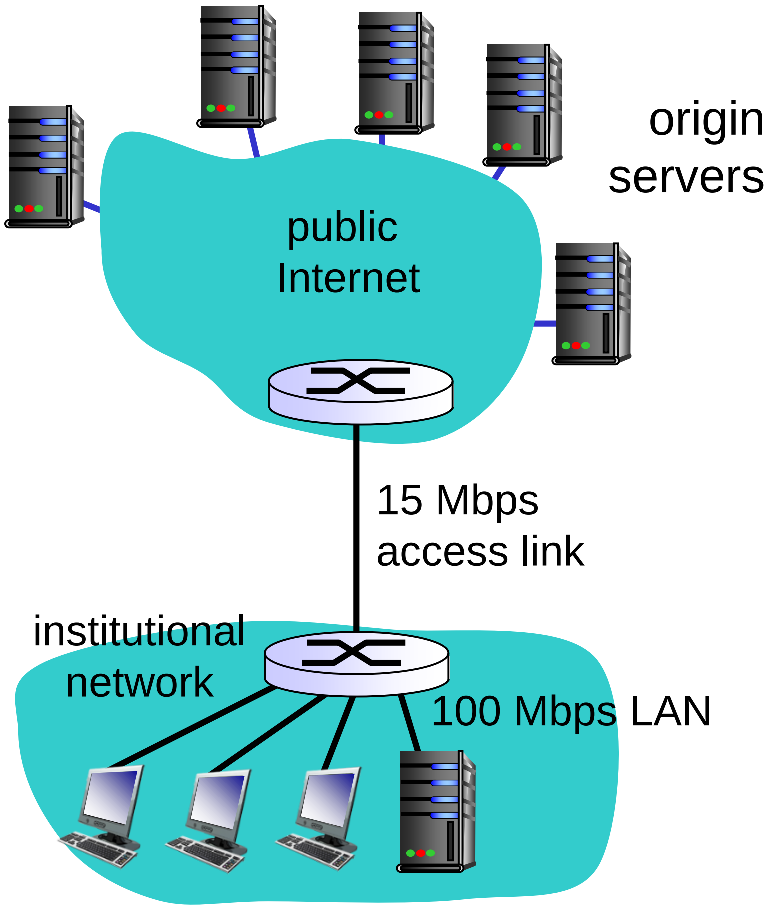
- **Web cache (Proxy server)**
  - Store recently requested object in its storage
    - Satisfy client request without involving origin server
  - Client (browser) send HTTP request to cache
    - Cache return object if it is in the cache
    - Cache request object from origin server if it is not in the cache
  - Advantage
    - Reduce response time for client request
    - Reduce traffic
    - Enable effective content delivery



2.2 Caching Example (1/3)

Assumption:

- Internet delay = RTT from internet side of public Internet router to any origin server: 2 sec



Consequences:

- Total response delay

  = Internet delay + access delay + LAN delay

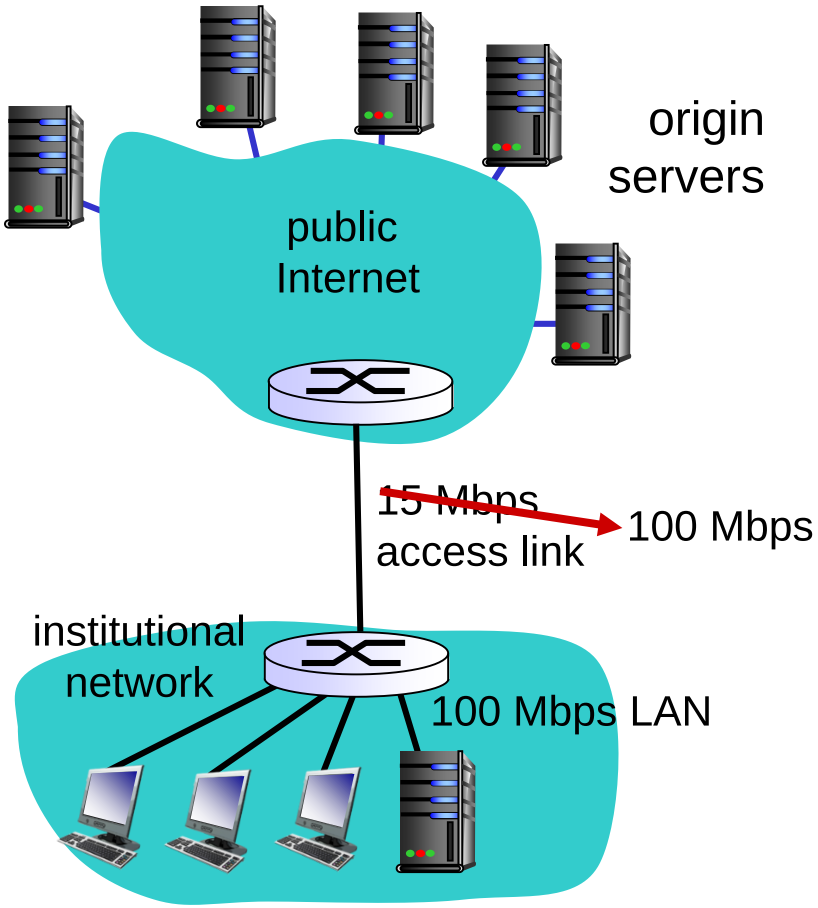
  = 2 sec + minutes + usecs

❖ Simple solution
  - Increase the access link speed

Consequences:

❖ Total response delay
  = Internet delay + access delay + LAN delay
  = 2 sec + usecs + usecs

❖ But, high cost to increase access link speed

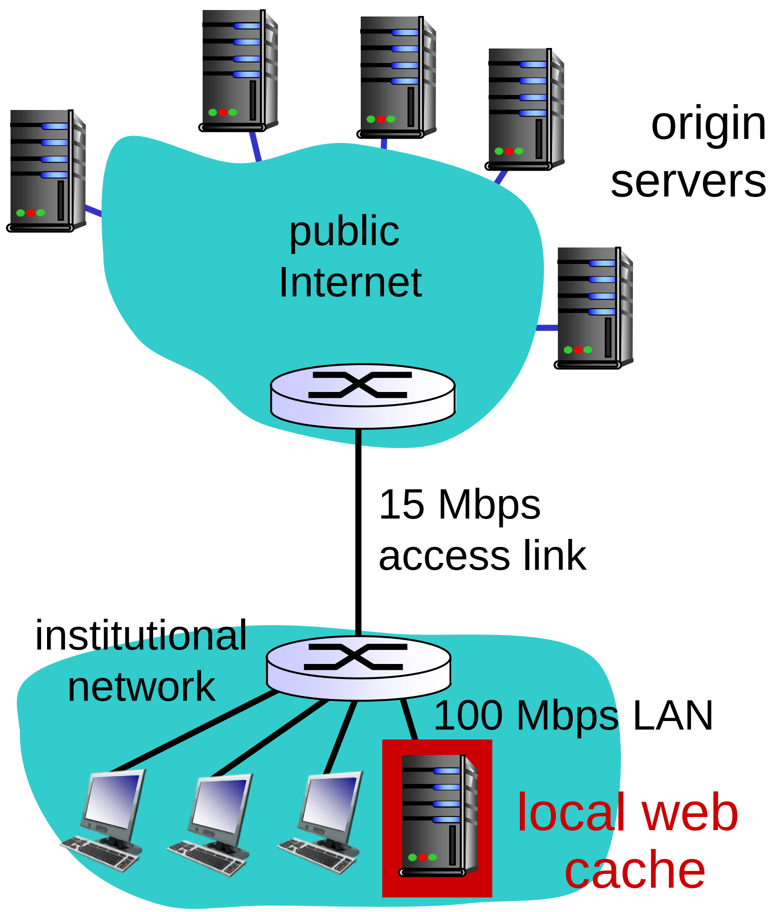


2.2 Caching Example (3/3): Install Web Cache

- Better solution
  - Setup local web cache

Consequences:

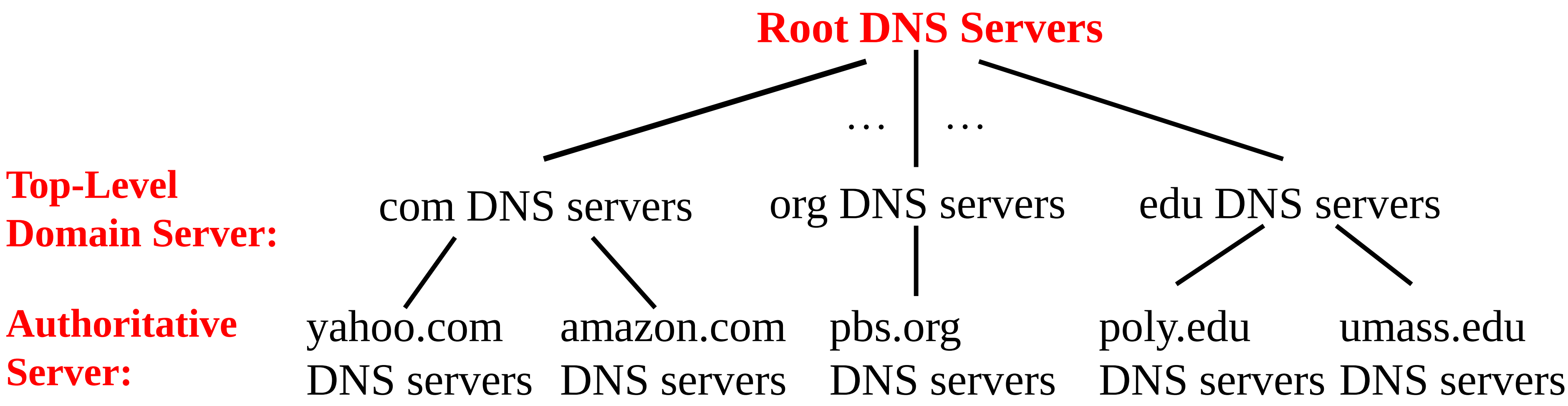
- Suppose cache hit rate is 0.4
  - 40% requests satisfied at cache
    - Incurs 0.01s
  - 60% requests satisfied at origin
    - Incurs 2.01s
- Total response delay
  $= (0.4 \times 0.01s) + (0.6 \times 2.01s)$
  $= 1.21s$



SUNWAY UNIVERSITY

- Main function
  - DNS server translate hostname to IP address
    - E.g.: cnn.com = 121.7.106.83
- Characteristic
  - A hierarchy of DNS servers as distributed database
    - Why not centralized?
      - Single point of failure
      - High traffic volume
      - High delay
        - DNS server may be far from DNS client
      - High maintenance
        - Large record for all hostname

2.4 Overview of How DNS Works: A Distributed, Hierarchical Database



Client wants IP address for hostname www.amazon.com

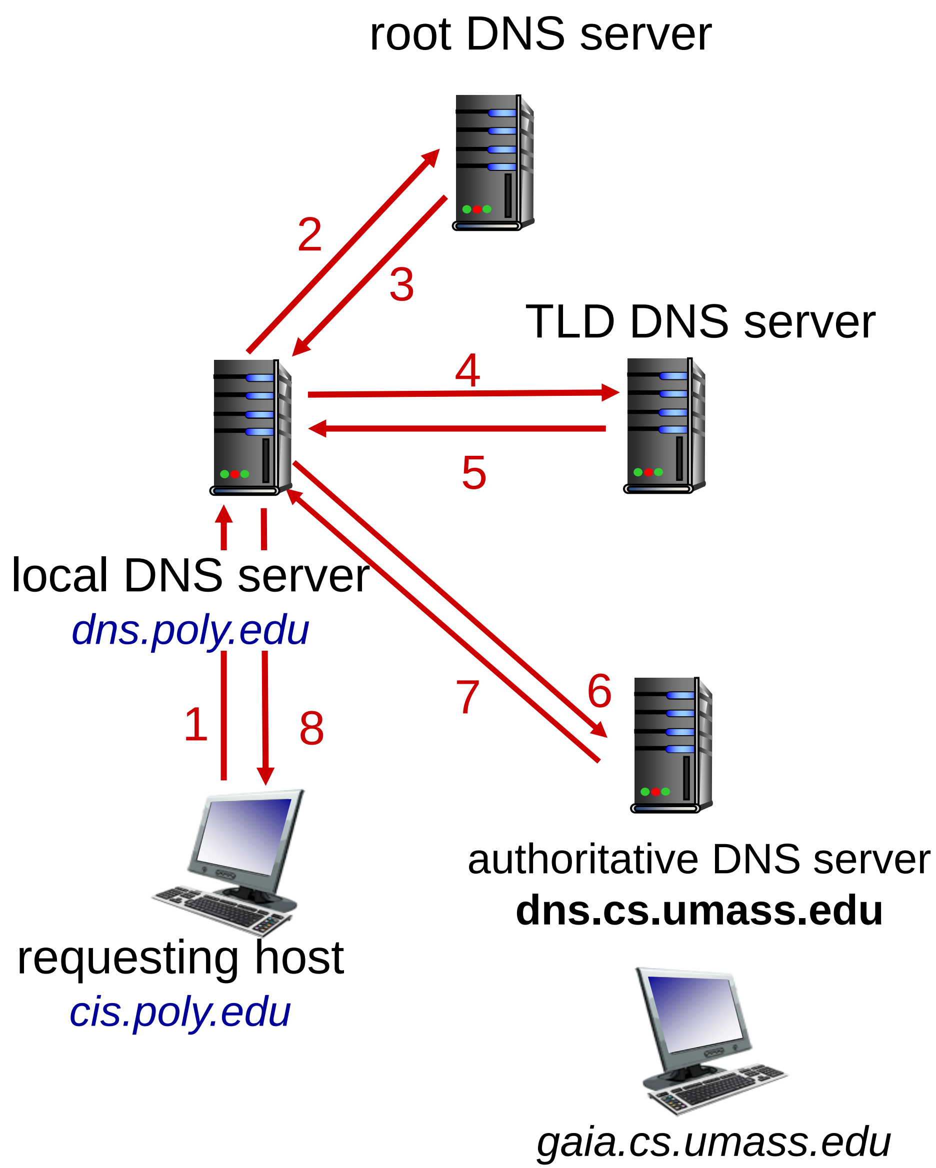
- Client queries **root DNS server** to find com DNS server (**Top-Level Domain server, TLD**)
- Client queries com DNS server to get amazon.com DNS server (**Authoritative server**)
- Client queries amazon.com DNS server to get IP address for www.amazon.com

2.4 Overview of How DNS Works: A Distributed, Hierarchical Database

- Host at cis.poly.edu wants IP address for gaia.cs.umass.edu
- Two types of query
  - Iterated query
  - Recursive query

Iterated query

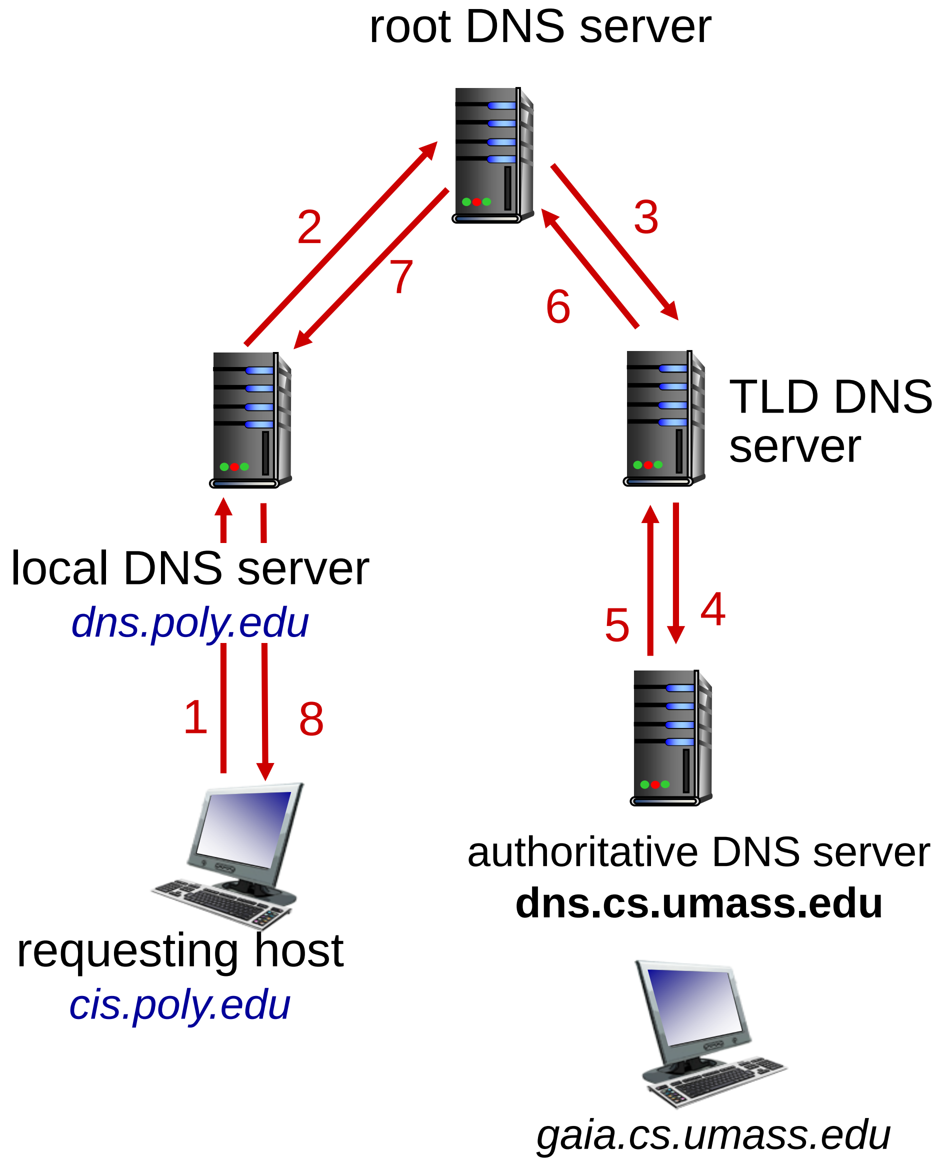
- Contacted server reply with name of server to contact
- "I don't know this name, but ask this server"



2.4 Overview of How DNS Works: A Distributed, Hierarchical Database

Recursive query

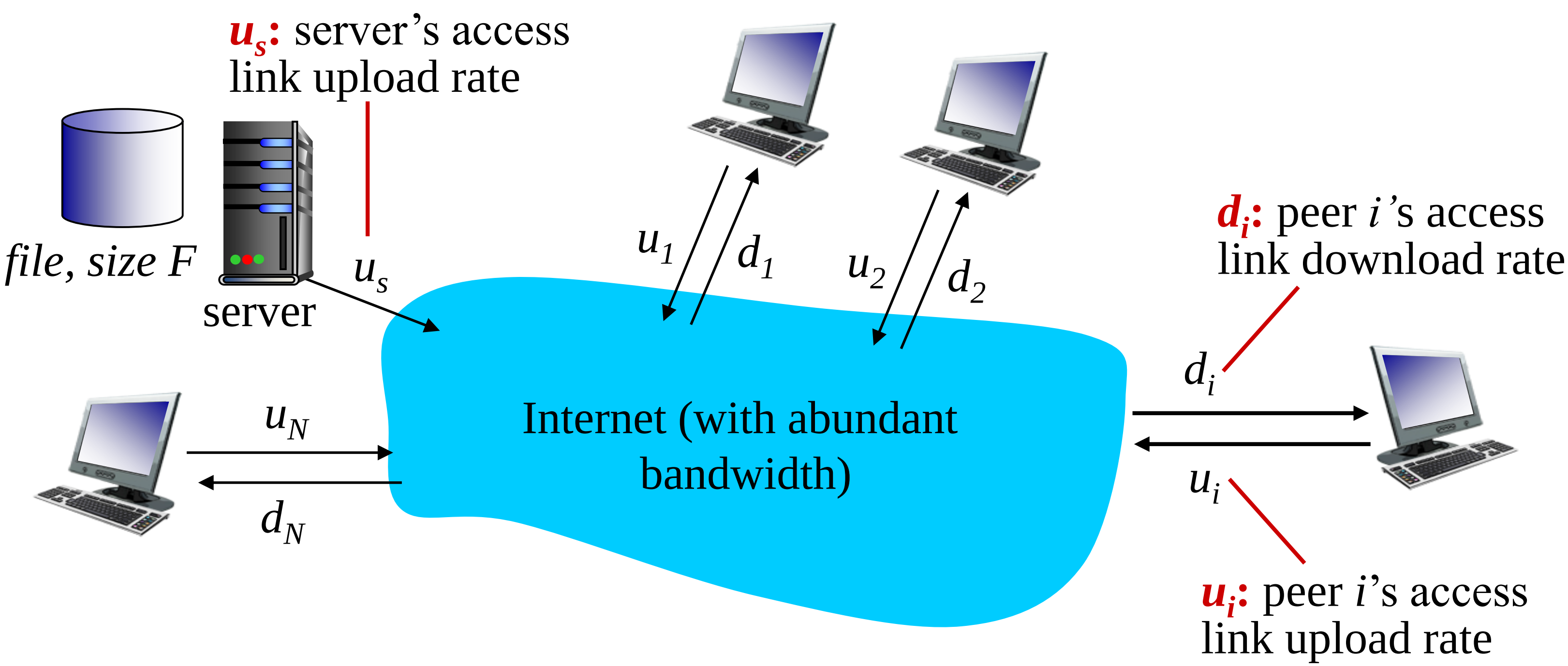
- Puts burden of name resolution on contacted name server
- Heavy load at upper levels of hierarchy



- **DNS caching**
  - Objective
    - Reduce number of DNS message
    - Improve delay performance
  - Function
    - Local DNS server
      - Store mapping of hostname and IP address in cache
      - Retrieve cached IP address
        - So, reduce visits to root name servers
  - Cache entry timeout (disappear) after some time (e.g. two days)
    - So, out-of-date cache entry is removed

- Refer to Slide 4

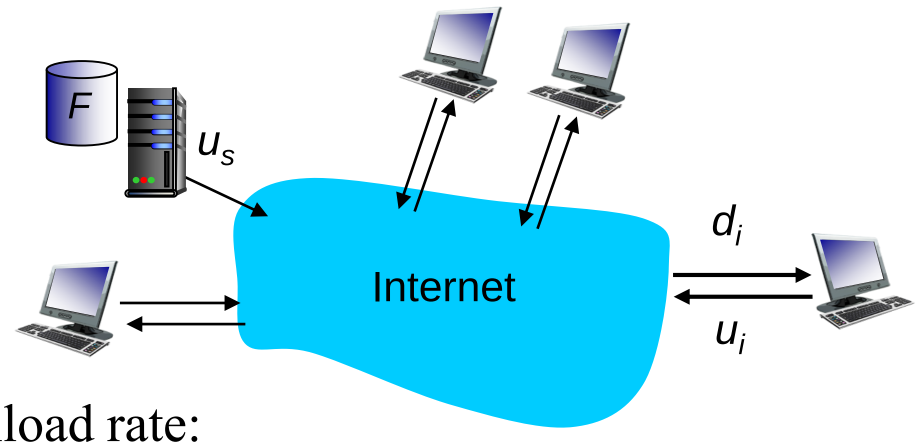
Question: Given that peer upload and download rate are limited resources, what is the distribution time to distribute file (size *F* bits) from one server to *N* peers?



2.5 Peer-to-peer Applications: Distribution Time for Client-Server Architecture

- Server
  - Send (upload) N copies of file sequentially
  - Time to send one copy: F/u_s
  - Time to send N copies: NF/u_s
- Client
  - Download a copy of the file
  - Minimum client's access link download rate:

    $d_{\text{min}} = \min \{d_1, d_2, \ldots, d_N\}$
  - Minimum client download time: F/d_{min}



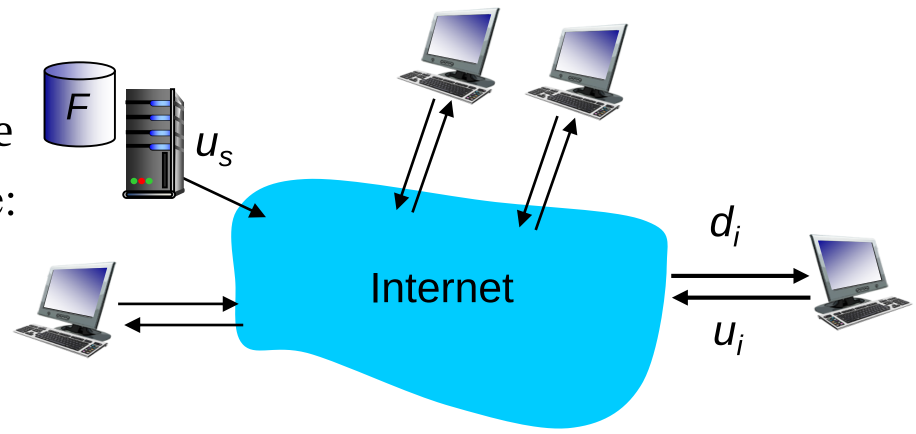
Distribution time to distribute file (size F bits) to N clients using client-server approach

$$D_{cs} = \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$

2.5 Peer-to-peer Applications: Distribution Time for Peer-to-peer Architecture

SUNWAY UNIVERSITY

- Server
  - Upload one copy of file
  - Time to send one copy: F/u_s
- Client
  - Each client must download a copy of the file
  - Minimum client's access link download rate: $d_{min} = \min\{d_1, d_2, \ldots, d_N\}$
  - Minimum client download time: F/d_{min}
  - Total upload rate of server and all peers: $u_{total} = u_S + u_1 + \ldots + u_N$
  - Minimum client download rate with peer upload: NF/u_{total}



Distribution time to distribute file (size F bits) to N clients using peer-to-peer approach

$$D_{P2P} = \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_{total}}\right\}$$